17:27:03 OCA PAD INITIATION - PROJECT HEADER INFORMATION 05/23/88

Active

Project #: E-24-645 Cost share #: E-24-359 Rev #: 0
Center # : R6497-0A0 Center shr #: F6497-0A0 OCA file #:
Work type : RES
Contract#: 19X-SB778C Mod #: Document : SUBCONT
Prime #: DE-AC05-840R21400 Contract entity: GTRC

Subprojects ? : N
Main project #:

Project unit: ISYE Unit code: 02.010.124
Project director(s):
JARVIS J J ISYE

Sponsor/division names: OAK RIDGE NAT'L LAB / MARTIN MARIETTA
Sponsor/division codes: 240 / 001

Award period: 880401 to 881231 (performance) 881231 (reports)

Sponsor amount	New this change	Total to date
Contract value	100,342.00	100,342.00
Funded	100,342.00	100,342.00
Cost sharing amount		10,000.00

Does subcontracting plan apply ?: N

Title: ENHANCEMENT AND VALIDATION OF ADANS AIRLIFT LOAD PLANNING

PROJECT ADMINISTRATION DATA

OCA contact: John B. Schonk 894-4820

Sponsor technical contact

FRANK SOUTHWORTH

MARTIN MARIETTA ENERGY SYSTEMS
P.O. BOX M
OAK RIDGE, TN 37831

Sponsor issuing office

JOHN E. SCHULTZ

SAME

Security class (U,C,S,TS) : U ONR resident rep. is ACO (Y/N): N
Defense priority rating : supplemental sheet
Equipment title vests with: Sponsor X GIT

Administrative comments -
PROJECT INITIATION
THIS PROJECT IS ADMINISTERED UNDER USUAL MARTIN MARIETTA/OAK RIDGE T'S & C

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Date 7/31/89

ject No. E-24-645 Center No. R6497-0A0

ject Director J. J. Jarvis School/Lab ISyE

nsor ORNL/Martin Marietta Energy Systems, Inc.

tract/Grant No. 19X-SB778C GTRC XX GIT ___

ne Contract No. DE-AC05-840R21400

le Enhancement and Validation of Adans Airlift Load Planning

ective Completion Date 3/31/89 (Performance) 3/31/89 (Reports)

eout Actions Required:

None
Final Invoice or Copy of Last Invoice
Final Report of Inventions and/or Subcontracts - Patent Questionnaire sent to PI
Government Property Inventory & Related Certificate
Classified Material Certificate
Release and Assignment
Other ___

udes Subproject No(s). ___

roject Under Main Project No. ___

inues Project No. ___ Continued by Project No. ___

ribution:

Project Director
Administrative Network
Accounting
Procurement/GTRI Supply Services
Research Property Management
Research Security Services

x Reports Coordinator (OCA)
X GTRC
X Project File
2 Contract Support Division (OCA)
___ Other ___

PDRC Report Series 88-10
November 1988
Revised: March 8, 1989

VALIDATION OF AIRCRAFT LOAD PLANNING ESTIMATOR PDRC 88-10

by
J. Jarvis
H. D. Ratliff
B. Stutzman

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

This work was supported by the Martin Marietta Contract No. 19X-SB778C.

Contents

Abstract

This paper proposes validation steps for the Aircraft Load Planning Estimator (ALPE). Validation will include evaluating the sometimes competing objectives of algorithmic accuracy and runtime. The primary approach in validation will be comparisons of ALPE to established aircraft load planning algorithms using the same or similar sets of authentic aircraft cargo data. The proposed effort is over a three month period.

1 Introduction

The purpose of this paper is to propose methods for validating the earlier proposed Aircraft Load Planning Estimator (ALPE)[1] Emphasis in algorithm validation will be upon both the accuracy and the runtime involved in running ALPE, and on the trade-off between these two competing objectives. The primary approach in validation will be comparisons against existing algorithms for aircraft load planning. The databases of movement requirements will be realistic movement requirements supplied by MAC, manipulated in different ways to test the accuracy and runtime of ALPE.

The layout of the sections of this proposal are as follows: (1) a discussion of the data used for comparisons, (2) the other algorithms ALPE will be compared to, (3) a review of the experimental design for the comparison tests, and (4) a discussion of the possible benefit and cost functions used in deciding what cargo to pack on what planes.

2 Data

Oak Ridge National Laboratory (ORNL) has supplied Georgia Tech (GT) with level 2 TPFDD records representing a number of movement requirements (MRs). GT has the SRF and TUCHA files to recreate the levels 3 and 4 data corresponding to these records.

Level 2 records are general movement requirements containing information on total weight and volume. Movement requirements correspond to

[1] "Aircraft Load Planning Estimator." Production and Distribution Research Center, Georgia Tech. PDRC No. 88-09. October, 1988.

such things as battalions and companies. Level 3 records are more specific than level 2 records; they contain the same data as level 2 records plus square footage. Level 4 records contain information about unagreggated cargo. Level 4 items are individuals pieces of cargo like trucks and trailers. The level 4 information included is width, length, height, weight, and area.

Different subsets of the TPFDD database available will be taken (about 100-150 items in each subset) to create a file of the level 4 records pertaining to the TPFDD subset. The subsets will be selected so as to submit ALPE to a variety of different load requirements. This will allow better understanding of the average and worst-case performances of ALPE. The resulting files of level 4 records will have entries for the following (number, weight, length, width, weight, hazardous cargo key, outsize indicator, and priority (if any).

Georgia Tech will also attempt to create small cargo sets (enough to fill less than five planes) which will be "perfectly packed." In other words, Georgia Tech will try to test the limits of ALPE with cargo loads which by hand packing are known to fill a number of plane loads exactly. Not only will it look at the packings generated by ALPE but Georgia Tech will also check the ones by CALM (an interactive aircraft loader to be discussed later).

3 Algorithms Compared

ALPE will be compared to several other algorithms already developed for aircraft load planning. In some cases, the comparison will be solely to verify correctness or feasibility of the loads generated (i.e. CALM, a single-aircraft load planner). Other algorithms will not only be used to verify correctness but also for running time comparisons. In this case, verification of correctness means either comparison with the number of aircraft estimated by another algorithm or comparison with bounds suggested by another algorithm.

The comparisons will use the same movement requirements that ALPE will use. However, some algorithms use only level 2 data and some use levels 3 or 4 data. Therefore computing time for levels 2 or 3 algorithms cannot be directly compared to ALPE; accuracy comparisons to such algorithms can only be used in the establishment of bounds.

This section describes each of the algorithms considered. The order of presentation will flow from level 2 algorithms to level 4 algorithms.

3.1 Level 2 Algorithm

3.1.1 Preliminary Approach by Distinct

Distinct is currently employing algorithms with level 2 data to generate estimates of aircraft requirements. It will be necessary for Georgia Tech to inquire into the data formats required by Distinct's algorithm. Then, a few samples of 100-150 requirements will be sent to Distinct or ORNL. Distinct (or ORNL) will return the number of each type of aircraft estimated for each sample. These samples will be the same ones GT uses to generate its level 4 data.

3.2 Level 3 Algorithm

3.2.1 Previously Developed Algorithm by Georgia Tech

The algorithm developed in early 1988 by Georgia Tech worked with level 3 records. (See PDRC Report 88-03 for a description of this algorithm)[2] The same samples of level 2 requirements as earlier described will be used to generate level 3 data.

3.3 Level 4 Algorithms

3.3.1 CALM (A single aircraft packer)

Version 4.0 of CALM (Computer Aided Load Manifesting) has been made available to Georgia Tech for another purpose. It is available to generate aircraft load plans for the intention of comparing it to ALPE.

CALM takes a list of cargo together with an aircraft and attempts to create a feasible load on it. It is an interactive program which allows a load planner to move items around on a graphic depiction of the aircraft and cargo in order to create a good load.

[2] Jarvis, Ratliff, Hane, Stutzman. "Plane Loading Algorithms for Military Airlift Command." Production and Distribution Research Center, Georgia Tech, PDRC Report No. 88-03. 1988.

Since the algorithm only loads one aircraft at a time, and all of the level 4 records from the TUCHA and SRF will have to be translated manually into CALM recognizable records, CALM will only be used to verify small samples of movement requirements. Its primary use will be in verifying the feasibility of loads generated by ALPE. Again, due to the restriction of loading only one aircraft at a time, the sum of aircraft required by using CALM will provide an upper bound on the actual number of aircraft needed.

3.3.2 Work by Kirk Yost

Captain Kirk Yost (USAF) has recently completed work on his airlift estimation techniques[3] His work modified DMES (Deployable Mobility Execution System) to an algorithm based upon a modified first-fit bin- packing method. His algorithm provides bounds, like ALPE proposes to do, of the number of aircraft required, it also produces feasible first-cut load plans.

Continuing study will address whether it is possible to access his system. However, major problems exist in compatibilities with the computer system it runs on (Hewlett- Packard 9836 and 9816 microcomputers) and the format of files required (DMES deployment files). The evaluation will first attempt to use the same sets of data available for the Distinct and previous Georgia Tech algorithms. If it is not possible to use the exact data, an attempt will be made to run any comparable set of data using Yost's algorithm.

4 Experimental Design

The purpose of this section is to review the comparisons ALPE will be involved in. The last section primarily discussed the algorithms and the data sets used to make the comparisons. This section will describe experiments which "test the limits" of the ALPE algorithm (i.e. provide bounds on worst-case and best-case performances as well as define the realm of problems ALPE will work on).

[3]Capt. Kirk A. Yost, Ronald W. Hare. "Airlift Estimation for MAC Cargo Aircraft." Air Force Logistics Management Center Report No. LY870104. August 1988.

4.1 Data Sets

As discussed in sections 2 and 3, most of the movement requirement data will be formed from the TPFDD, TUCHA, and SRF files supplied to GT by ORNL. Data sets of aircraft will be formed totally from values obtained from papers describing the maximum ACL (available cabin load), length, width, etc. of the aircraft used.

The perfectly packed data set will not necessarily come from actual MR data sets but will be generated to test ALPE in circumstances where the cargo fit together exactly.

Movement requirement data sets will consist of about 100-150 level 2 items each (these directly correspond to TPFDD records). The corresponding levels 3 and 4 records will be formed from the level 2 records. Priority values will be obtained from the level 2 TPFDD records. These values are used by ALPE to decide which movement requirements to leave out if all MRs do not fit on the aircraft supplied.

Aircraft data sets will consist of C-5, C-141 and C-130 aircraft. Ordering of the aircraft will vary as well as remaining ACL (i.e. the aircraft could be partially filled when they arrive).

4.2 Testing Limits

Examination of the performance of ALPE with the two or three sample data sets used with other algorithms will lead to the creation of other data sets which will hopefully permit the characterization of a relationship between data and performance. Knowledge obtained in the original samples will help in form conjectures on how, for example, the ordering of the data and the range of cargo characteristics (amount of palletized, outsize, and/or hazardous cargo) affect performance. These conjectures will lead to the creation of data sets which will test them. Tests will also evaluate ALPE performance in situations where the number of aircraft supplied is/is not a constraining factor.

The results of these tests will hopefully aid in defining the characteristics of situations where ALPE works well and situations where it doesn't. They will also provide bounds on performance.

5 Cost and Benefit Function Tests

As mentioned in the earlier ALPE report (see footnote 1), where an item or strip (a pallet-wide length of items) is placed (i.e. which aircraft) depends on the results of the cost and utility functions. Benefit functions provide a sense of how well a strip fits on a certain partially-filled aircraft and cost functions provide a sense of how much "harm" can be done by putting a strip on the aircraft. These functions may or may not be of the same form.

All of these functions consider remaining aircraft capacity with respect to some or all of the constraining factors (i.e. length, weight, etc.) and the characteristics of the strip in question. The difference among the functions is in how these elements are considered. For example, the proposed utility function in the previously referenced report on ALPE is

$$U(\text{strip } i) = \min\{1/RL_{ij}, 1/RACL_{ij}\}$$

where RL_{ij} is the remaining length of the cargo bay of aircraft j after strip i is added,

and $RACL_{ij}$ is the remaining ACL.

Other functions could, for example, (1) use logarithms instead of reciprocals of the constraining factors, (2) combine the factors in other ways, possibly squaring and then adding them, instead of taking the minimum.

At least three different utility functions will be used and tested. A factorial design with each of the proposed functions and different sets of MRs and aircraft will be performed in order to decide which functions work best in particular situations; and which function works well in most situations.

6 Review

The following figure summarizes the validation phase of the PDRC Aircraft Load Planning Estimator Effort.

Steps	Scheduled Completion

1. Jan.

a) Identify at least two cargo sets which will perfectly pack about five planes each.

b) Identify at least three sets of 100-150 MRs (level 2).

c) Form the corresponding levels 3 and 4 data.

d) Identify a set of aircraft to be used (one type, all empty, more than enough to load all items).

2. Early Feb.

a) Submit the perfectly packed sets to CALM and ALPE

b) Submit the level 2 data to Distinct

c) Submit the level 3 data to old PDRC algorithm

d) Submit the level 4 data to ALPE, possibly Yost algorithm

e) Submit subset of level 4 data to CALM

3. Late Feb.

a) Compare number of aircraft estimated, time of computation, and loads of the above.

b) Identify possible areas for future testing of ALPE algorithm (e.g. more outsize items, more hazardous cargo, different aircraft mixes).

4. Early March

a) Examine results and form conjectures on where ALPE works well, where it doesn't, expected computational time.

5. Early March

a) Identify two or three different cost and benefit functions for testing.

b) Identify three or four sets of level 4 data including some small perfectly packed sets.

c) Identify two or three sets of aircraft.

6. Mid March

a) Run 2(or3)X3(or4)X2(or3)level factorial design

b) Identify cost and benefit functions where ALPE works best

7. End March

a) Produce report, user manual, etc.

Figure 1: Proposed Steps for Validation of ALPE Algorithm

PROGRESS REPORT

Georgia Tech Aircraft Loading Project

August 1988

Activities Performed:

Accomplishments

In August preliminary algorithmic development started. The algorithm is a three-phase approach. The first phase is to sort and separate the cargo items. The second phase is to pack the cargo into strips, one plane width wide. Next, the strips are collected into plane loads using a multi-constraint knapsack heuristic. The algorithm is currently being coded on the IBM 9375.

Verification

Examination of current plane loading routines (i.e. **CALM**) is under way. This effort is required so that comparisons of our algorithm to a benchmark can be performed.

PROBLEMS

In addition to all of the information requested, and trouble spots pointed out in last month's report, several other issues need clarification.

Partially Loaded Aircraft

When aircraft arrive partially loaded, how much freedom exists to re-arrange those partial loads? If partial loads cannot be re-arranged, then the heuristic solution will suffer. If re-arrange-

ments are allowed the computation of the plane loads could require a lot of computer time.

Without answers to previously posed questions, the algorithmic development at Georgia Tech may be hampered and delayed.

Assumptions

So that research may continue, we are assuming that one aircraft type is passed to our subroutine. The number of these aircraft is also specified. Each aircraft has its own cargo bay length, and ACL (to allow for partially packed planes); but they all share a common width. This is necessary for packing cargo in plane width strips.

The algorithm may be flexible enough to handle either objective criterion mentioned in the July report. In lifting all the cargo, the number of planes passed to the subroutine is considered as the number of partially packed planes arriving, and there exists an additional pool of empty planes from which we can draw. In dealing with prioritized cargoes, the number of planes passed is considered an absolute upper bound. The cargo lifted is that which in some sense maximizes the value of the loads.

Data

Incorporation of the data sent on August 10, 1988 has begun. Documentation was received for the UNIT and NONUNIT data files. However, there was no documentation received for LEGCONV and REQCONV files. It was welcome news that the UNIT and NONUNIT data files followed standard **TPFDD** conventions. But, TPFDD's only carry

Level 2 data and not our required Levels 3 and 4.

E-24-64

September 1988 PDRC Report

Accomplishments

Finished Report About to be sent is a report on the PDRC **Airlift Load Planning Estimator** (ALPE). An algorithm, ALPE, is described which uses Level 4 input of cargo and an aircraft list to output the number of each plane type used and their remaining cargo space and any cargo which could not be loaded. A combination of bin-packing and multi-constraint knapsack algorithms are to be used in ALPE development.

Report in Progress Soon to be completed is a report on the validation of ALPE on realistic data. Validation effort is to be focused not only on accuracy but on speed of computation. PDRC requirements for realistic data will be mentioned in the report. The report will also mention "benchmark" algorithms such as CALM to which ALPE will be compared for the reasonableness of loads generated.

ORNL, Distinct Reports We have received and have examined the copies of the ORNL "Progress Report for the ADANS Execution Planning Algorithm" and Distinct "Progress Report for the ADANS Deliberate Planning Algorithm." They are much appreciated. Discussed in the next section is our continued muddled understanding of where the PDRC effort fits in.

Continuing Problems

Where PDRC effort fits in and our assumptions After studying the Distinct report, it appears that the PDRC effort either **is** or **is part of** the PKER subroutine ("This subroutine will pack a plane type with the cargo of a requirement", p. 32). Our inclusion in the PKER subroutine is not obvious. Other assumptions and problems are listed below. Explicit mention of our effort is only made on page 81 of our "load planning capabilities."

Input format We are continuing to presume that we are passed a list of available planes of one aircraft type with the length and ACL constraints of each. The format of such a file is not known. Level 4 cargo data is in the form of SRF detail cargo records. This assumption has not been verified.

Output format It is assumed that we will return the number of planes required and will update the plane and cargo list to show remaining length and ACL of the aircraft and the cargo not loaded.

Files received on August 10, 1988 We have still not received documentation for either the LEGCONV or REQCONV files. None of the four files (previously mentioned two plus UNIT and NONUNIT) appear to carry Level 3 or 4 cargo data or aircraft data.

E-24-645

October 1988 PDRC Report

Accomplishments

Finished Report About to be sent is a report on the validation of the PDRC **Airlift Load Planning Estimator** (ALPE). Validation includes comparisons to existing routines for aircraft load planning (CALM, Distinct preliminary approach, previous PRDC work, Kirk Yost work). These comparisons will look at accuracy and time of computation. Effort will also be performed to "test the limits" of ALPE. The report mentions the work by Kirk Yost. We would like to see if we can access his system.

Preliminary tests of ALPE We have successfully tested ALPE with simple data. About six plane loads of level 4 items were packed. The data was not obtained directly from the TUCHA and SRF files but was created separately using fairly realistic sized cargo. Work still needs to be done in manipulating prioritized cargo and in situations where the number of aircraft supplied is not enough to load all cargo.

Continuing Problems

This section is the same as in the September 1988 report except as noted.

Where PDRC effort fits in and our assumptions After studying the Distinct report, it appears that the PDRC effort either is or is part of the PKER subroutine ("This subroutine will pack a plane type with the cargo of a requirement", p. 32). Our inclusion in the PKER subroutine is not obvious. Other assumptions and problems are listed below. Explicit mention of our effort is only made on page 81 of our "load planning capabilities."

Input format We are continuing to presume that we are passed a list of available planes of one aircraft type with the length and ACL constraints of each. The format of such a file is not known. We are now assuming (different than in Sept. 88 report) that level 2 cargo data is supplied in the form of TPFDD cargo records. Level 4 cargo information is then formed by our algorithm using TUCHA and SRF data files.

Output format It is assumed that we will return the number of planes required and will update the plane and cargo list to show remaining length and ACL of the aircraft and the cargo not loaded. The format of such files is unspecified as of yet.

GEORGIA INSTITUTE OF TECHNOLOGY

PROJECT EXPENDITURE & BUDGET REPORT

PRINCIPAL INVESTIGATOR J J JARVIS CENTER NO. 246R6497OAO ACCOUNT NO. E-24-645

TRANSACTIONS FOR OCTOBER 1988 DEPARTMENT I & S ENG

DATE	DESCRIPTION	DOCUMENT ID/REFERENCE		CODE	BUDGETS	ENCUMBRANCES	EXPENDITURES
PERSONAL SERVICES							
10/31/88	HANE,CHRISTOPHER			51144			782.66
10/31/88	JARVIS,JOHNJ			51110			1,952.09
10/31/88	STUTZMAN,BRYANR			51144			782.66
10/12/88	ENCUMBRANCE ADJSTMNT		10-P016	C 51144		4,695.96	
10/31/88	MONTHLY PAYROLL		10-P045	C 51110		-1,952.09	
10/31/88	MONTHLY PAYROLL		10-P045	C 51144		-1,565.32	
	TOTAL					1,178.55	3,517.41
FRINGE BENEFITS							
10/31/88	OCT 88 SPON BENE		10-P053	C 52024			497.78
10/21/88	SPON BENEFITS ENCUMB		10-P032	C 52024		-497.78	
	TOTAL					-497.78	497.78
MATERIALS AND SUPPLIES							
10/27/88	JOHN WILEY & SONS	240900416	00520064	M 72760			75.00
10/27/88	PPC CHGS 10/88 1527		10-033	C 74250			2.55
10/26/88	JOHN WILEY & SONS	240900416		H 72760		75.00	
10/27/88	JOHN WILEY & SONS	240900416	00520064	M 72760		-75.00	
	TOTAL					.00	77.55
OVERHEAD							
	TOTAL RATE OF 60.0%	BASE OF 4				408.46	2,455.64
BUDGET CHANGES							
	PERSONAL SERVICES				4,696.00		
	OVERHEAD				2,818.00		
MONTHLY	TOTAL					1,089.23	6,548.38

OVERHEAD COMPUTED ON TOTAL DIRECT CHARGES MINUS EQUIPMENT, SUBCONTRACTS OVER AUTHORIZED BASE, AND STUDENT AID PAGE 01

GEORGIA INSTITUTE OF TECHNOLOGY

PRINCIPAL INVESTIGATOR	J J JARVIS	CENTER NO.	246R64970A0	ACCOUNT NO.	E-24-645
STATUS AT END OF	OCTOBER 1988			DEPARTMENT	I & S ENG
SPONSOR	MARTIN MARIETTA ORNL				
AWARD NUMBER	19X-SB778C	RF CENTER NO.	00646073000	RESTRICTED FUND	RF-49148
EFFECTIVE DATE	04-01-88	BILLING GROUP	GTRC	EXPIRATION DATE	12-31-88

	MONTH	FISCAL YEAR	TOTAL CONTRACT
PERSONAL SERVICES			
BUDGET			51,789.00
EXPENDED	3,517.41	20,782.11	28,779.57
ENCUMBERED	1,178.55	7,034.82	7,034.82
FREE BALANCE			15,974.61
FRINGE BENEFITS			
BUDGET			10,971.00
EXPENDED	497.78	3,728.30	5,287.56
ENCUMBERED	-497.78	995.57	995.57
FREE BALANCE			4,687.87
MATERIALS AND SUPPLIES			
BUDGET			900.00
EXPENDED	77.55	993.38	993.38
ENCUMBERED	.00	.00	.00
FREE BALANCE			-93.38
TRAVEL			
BUDGET			3,750.00
EXPENDED	.00	.00	.00
ENCUMBERED	.00	1,036.00	1,036.00
FREE BALANCE			2,714.00
TOTAL DIRECT CHARGES			
BUDGET			67,410.00
EXPENDED	4,092.74	25,503.79	35,060.51
ENCUMBERED	680.77	9,066.39	9,066.39
FREE BALANCE			23,283.10

PRINCIPAL INVESTIGATOR J J JARVIS CENTER NO. 246R64970A0 ACCOUNT NO. E-24-645

STATUS AT END OF OCTOBER 1988 DEPARTMENT I & S ENG

SPONSOR MARTIN MARIETTA ORNL

AWARD NUMBER 19X-SB778C RF CENTER NO. 00646073000 RESTRICTED FUND RF-49148

EFFECTIVE DATE 04-01-88 BILLING GROUP GTRC EXPIRATION DATE 12-31-88

	MONTH	FISCAL YEAR	TOTAL CONTRACT	
OVERHEAD				
BUDGET			40,446.00	
EXPENDED	2,455.64	15,302.26	21,036.29	RATE OF 60.0 BASE OF 4
ENCUMBERED	408.46	5,439.83	5,439.83	
FREE BALANCE			13,969.88	
TOTAL				
BUDGET			107,856.00	
EXPENDED	6,548.38	40,806.05	56,096.80	
ENCUMBERED	1,089.23	14,506.22	14,506.22	
FREE BALANCE			37,252.98	

E-24-645

November-December 1988 PDRC Report

Accomplishments

Testing of ALPE We have successfully tested ALPE with TUCHA data. Throughout this period various bugs in the ALPE system were diagnosed and removed. Three penalty and priority functions were tested. The different combinations, while yielding nearly equal numbers of planes, varied in the individual loads. This indicates that extensive testing may be necessary to distinguish among them. The results from these early tests were presented to Oak Ridge National Lab in our December meeting.

Next Quarter's Goals

December Meeting The December meeting between PDRC and ORNL allowed much needed discussion of areas for further enhancement. Chief among these areas are the development of another benefit function, lower and upper bounds on solution quality, option codes for selection of penalty, benefit, and cost functions, and the testing of perfectly hand packed data sets. Also, the use of Ada as an alternative language to FORTRAN was discussed.

Problems

A copy of the hazardous loading regulations has not yet been received. Also, are the only distinctions between safe, nuclear, chemical, and explosives; or must the cargo designated as security, or hazardous by table T-18, appendix T to JCS Pub 6, volume ii, Part 11, Chapter 1, be handled specially ?

In creating example data sets from the TUCHA and SRF, how many records should be used per set, (100 ,500 ,1000 ,5000) ? The subdivision of the data will allow a more complete analysis of the empirical behavior of the algorithms. Similarly, what criteria should be used to determine the number of planes passed for the test runs ? Should the runs be executed so that all items are packed, or use the weight of cargo to determine the number of planes ?

In the TUCHA not all bulk items are palletized. The PDRC was informed that all bulk items would be handled as pallets. Since this is not true of the TUCHA data, a volume estimate of the number of pallets required for each TUCHA record will be required.

October 10, 1988

AIRCRAFT LOAD PLANNING ESTIMATOR

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205

Contents

1 Introduction

This report presents an algorithm, ALPE (Aircraft Load Planning Estimator), for determining the number of aircraft required to lift a given set of cargo. The algorithm uses Level 4 data as input, specifically, length, width, height, weight, and cargo category codes present in SRF and TUCHA records. The algorithm reports the number of each plane type used, amount of usable space remaining on each of the aircraft, and any cargo which was not moved.

Many areas of improvement over current load estimation algorithms are implemented in this algorithm. Cargo integrity is maintained through every step of the algorithm, i.e. no item is assigned to more than one aircraft. Hazardous cargo items are also identified and handled separately. Also, optimization techniques play a role in assigning cargo to aircraft.

Based on previous Georgia Tech research for ORNL, the methodology used to implement this algorithm is a combination of bin-packing and multi-constraint knapsack algorithms. The structure is three parts: preprocessing, assigning individual items to strips one plane width wide, and combining strips to form aircraft loads. Assignment of items to strips is a bin-packing problem. The subroutine to perform this algorithm is called WIDTH- FIT. Combining strips to form aircraft loads can be performed by a multi-constraint knapsack heuristic, it will be referred to as LOADER.

1.1 Model

A convenient model for packing one type of empty plane is the following. Initially, begin with an unbounded rectangle whose width is that of the cargo bay. Items should be placed in this rectangle so that the length of the rectangle containing all of the items is minimized. However, the packing should also be decomposable into units that do not exceed the ACL of the plane type or its cargo bay length. The decomposability can be achieved by a level-oriented packing heuristic with a side constraint on weight (WIDTH-FIT). Once cargo pieces are assigned to these units, called "strips", the strips must be collected together to form good plane loads.

Since all strips are one plane width wide, the only constraints for an aircraft load are the overall length and ACL. To assign strips to one empty plane, one could solve a binary integer program with a variable for each

strip, and two constraints, for length and weight. The objective of the program is to maximize the utilization of the aircraft, or minimize slack in the constraints. A similar type of problem is called a multi- constraint knapsack problem. Its formulation is discussed in section 2.4.3. There are fast heuristic procedures to solve this problem[1]. ALPE builds upon such a heuristic to handle multiple aircraft, partially packed aircraft and incompatible cargo (LOADER).

In general, the presence of hazardous cargo, outsize cargo and partially packed aircraft complicate the actual implementation of the above ideas. However, even with these complications the framework is robust enough to handle them. When outsize and oversize cargo are present, strips must be generated for the outsize cargo using a C-5 bay width, and for the other cargo using the width of the available plane type. If C-5's are available, and the outsize cargo does not fill the C-5, two regular width strips can be combined to form a C-5 strip.

[1]Loulou and Michaelides, New Greedy-like Heuristics for the Multidimensional 0-1 Knapsack Problem, Operations Research, vol. 27, no. 6, 1979

2 General Algorithm Structure

The initial stage of the ALPE algorithm is a pre-processing one. The individual handling requirements of the diverse types of cargo require that the cargo list be separated. Hazardous and outsize cargo are two examples of cargo classes that require special attention. In general, cargo is separated into the following classes: hazardous and outsize, outsize, hazardous and oversize, oversize, hazardous and bulk, and bulk. In addition to separation by classes, the pre-processing stage also sorts the cargos in the classes.

The WIDTH-FIT subroutine generates strips of cargo whose width is no more than the width of the cargo bay. This subroutine is executed for each of the non-bulk cargo classes. Bulk cargo is assumed to be palletized, and therefore either one or two pallets form a strip. As the individual cargo classes require different loading techniques, so also do their strips.

The LOADER subroutine combines strips to form aircraft loads. Thus it must be executed for each different width of strips generated by WIDTH-FIT. If all planes were C-141's or C-130's only one pass of LOADER is necessary. However if outsize items along with other cargo are present, two passes through LOADER must be executed.

2.1 Inputs to ALPE

There are two input lists that ALPE uses. The first is the cargo list. This list must include the information present in the SRF Force Cargo Detail Record. Among the data in this record, the important fields are the cargo category code, cargo dimensions, weight, and number of pieces. Optionally, priorities may be assigned to each item.

The second list provided to ALPE is the available aircraft list. This list must specify the aircraft type, including configuration if necessary, aircraft capacities, and number of aircraft that share these properties. It is allowable that the aircraft on the list be partially loaded. In this case the aircraft capacities are the remaining capacities, and there must be a flag indicating whether hazardous cargo is on board.

2.2 Pre-Processing

Due to the variety of item sizes and possible incompatible cargo types, pre-processing the data is necessary for guaranteeing good performance in WIDTH-FIT. Theoretical results for bin-packing algorithms indicate that sorting the data can have a profound affect on algorithmic efficiency[2]. The discrepancies of cargo types: outsize, bulk, hazardous, etc., require individual treatment for loading. Therefore dimensional data for these types are collected separately.

2.2.1 Hazardous Cargo

Hazardous cargo, those pieces whose first position cargo category code is either D, E, K or L (see table T-18 in Appendix T), require special placement on board an aircraft. To mimic this placement restriction, all dimensions of hazardous items are stored in an array separate from the other data. After the hazardous items have been placed in plane width strips by WIDTH-FIT, each strip is assigned to a unique plane. This insures that the amount of hazardous material placed on any one plane is limited.

2.2.2 Outsize Cargo

Another class of cargo that requires separate handling is outsize cargo. Outsize cargo is defined as any air transportable cargo which requires the use of a C-5 aircraft. Since outsize cargo requires a specific plane type, all dimensional data for outsize cargo is stored in a separate array. Also, cargo that is both hazardous and outsize, is separated from the rest of the cargo data.

2.2.3 General Cargo

To increase the efficiency of the bin-packing subroutine, the data must be ordered by non-increasing length. WIDTH-FIT is similar to Coffman, et al's Next-Fit Decreasing Height (NFDH) heuristic. Since ALPE performs packings on a horizontal surface (the cargo bay), the dimensions of concern are length and width, where length corresponds to height in the literature.

[2]Coffman, E.G., et al., Performance Bounds for Level-Oriented Two-Dimensional Packing Algoritms, SIAM J. of Comput., vol. 9, no. 4, Nov 1980, pp. 808-826

In the NFDH heuristic rectangles are packed left-justified on a level until there is no more room to the right to allow the next rectangle to fit. When this occurs, a new level is defined, the old level is closed, and packing continues on the new level. If item length is strongly correlated with its width, then sorting by decreasing length, also tends to order the items by width. This causes problems for the NFDH heuristic. An example of a NFDH and a WIDTH-FIT packing is shown in Figure 2.2.3.

For example, if many of the items have width exceeding one- half the width of the cargo bay and these items are adjacent in the input list, NFDH will pack only one item per level. In this situation it would be advantageous to examine more than just the next item on the list to try to find an item that will fit next to the wide item.

This argument is the motivation for separating the cargo into lists of items that exceed one-half the width of the cargo bay and the narrower items. Thus, all compatible cargo, i.e. hazardous and outsize, outsize, hazardous, and non-hazardous, are partitioned into wide and narrow lists. Then when the items are being assigned to strips, a wide item is placed on the strip then the narrow list is searched for items that will fit adjacent to it. How this is actually performed will be discussed in the next section.

The output of the pre-processing stage is a list of wide items and a list of narrow items for each compatible non-bulk cargo class. These classes are: hazardous and outsize, outsize, hazardous and oversize, and oversize. The two other cargo classes, hazardous bulk and non-hazardous bulk, are assumed to be palletized and have a fixed width. All cargo class lists are ordered by non-increasing length.

If the items in Figure 2.2.3 are all members of the same cargo class, then items 1 and 2 are in the wide list, and 3 through 7 are in the narrow list. The WIDTH-FIT packing in Figure 2.2.3 shows three strips, the strips contain items 1 and 4, items 2,5 and 6, and items 3 and 7.

2.3 Width-Fit

2.3.1 Introduction

This stage of the ALPE system assigns each item to a strip, a rectangle the width of the aircraft whose length is determined by the longest item assigned to the strip. This subroutine is executed once for each of the

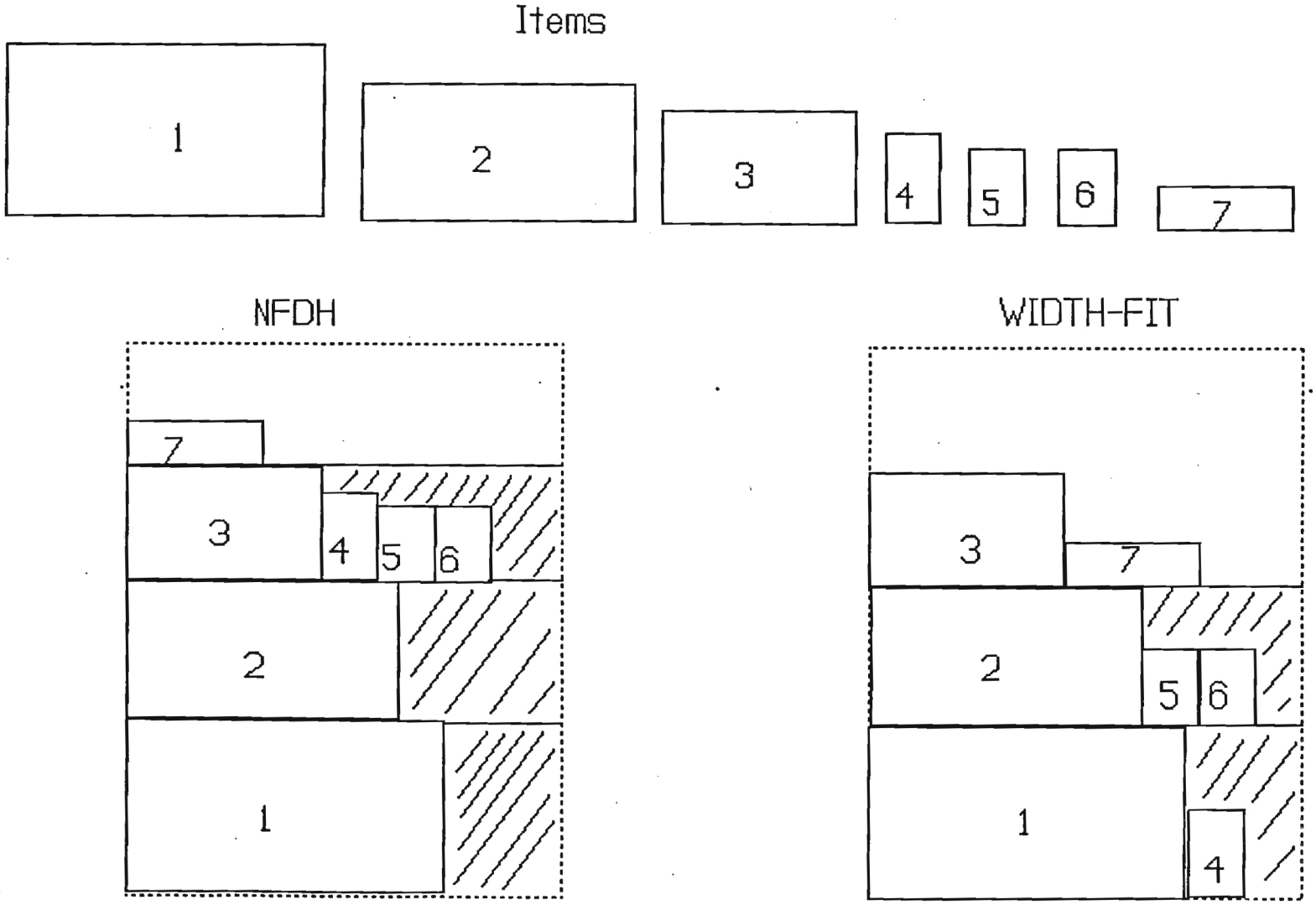

Figure 1.

cargo classes described in the previous section. It is necessary to separate the strips generated for each cargo class because the strips in the hazardous classes require special attention, and those in the outsize classes require a C-5 aircraft. This section will describe how WIDTH-FIT assigns items to strips.

2.3.2 Inputs

There are three main inputs to WIDTH-FIT, the wide and narrow item dimension lists and the plane capacity list. The wide and narrow item dimension lists are ordered by non-increasing length. All of the aircraft passed to WIDTH-FIT in one call must share the same cargo bay width. Thus, strips for C-141's together with strips for C-130's can be generated in one pass, but strips for C-5's and C-141's cannot. However, two strips for a C-141 could be combined to form a strip for a C-5. Implicit within the subroutine is the constraint that any item passed to the subroutine can be loaded onto any of the plane types present in the plane list.

2.3.3 Making Strips

The main body of WIDTH-FIT consists of two doubly-nested loops. The first loop iterates over the items in the wide item list. Only one wide item can fit on a strip, so each wide item is placed on its own strip. The inner loop iterates over the narrow item list.

Once a wide item is placed on a strip, the unassigned narrow items are checked to see if they fit in the remaining width and weight of the strip. If a narrow item is placed on a strip, the remaining width and weight of the strip are updated, the item is marked as packed, and the next item on the narrow item list is checked. After the remaining width or weight of the strip passes below a threshold, the strip is full and the loop can be exited. If the entire narrow item list is examined, the strip is also full, in the sense that no unpacked item can be assigned to it.

After all the wide items are assigned to strips, there are two possible situations to handle. The first is that there are no unpacked narrow items. In this case, WIDTH-FIT can terminate because all items are assigned to strips.

The second case is that there are some unpacked narrow items remaining. When this happens the second of the doubly-nested loops is entered. The procedure here is the same as that in the first double loop except that both loops iterate over the remaining narrow items. Thus, a narrow item is assigned to an empty strip, then the remaining unpacked narrow items are checked to see if they fit adjacent to that item on the strip.

2.3.4 Output

The output of WIDTH-FIT is the list of strips with their dimensions and a manifest of the items on that strip. Also, any item whose length or weight exceeds the maximum aircraft capacity of that dimension can immediately be classified as not movable. This is true since partially packed aircraft are not repacked. Again, in Figure 2.2.3 there are three strips. Strip 1 has length equal to that of item 1 and weight equal to the sum of the weights of items 1 and 4. Similarly, strip 2 has length equal to that of item 2 and weight equal to that of items 2, 5 and 6. The same holds for the third strip consisting of items 3 and 7.

2.4 Loader

2.4.1 Introduction

The purpose of LOADER is to take all the strips generated by the calls to WIDTH-FIT, and combine them to form good loads. If there are outsize items present, LOADER must be called twice, once for the outsize items and then for the rest of the cargo. LOADER consists of two stages.

2.4.2 Hazardous Cargo

In the first stage, each hazardous strip is assigned to a plane that contains no hazardous material. This is done to limit the amount of hazardous material on board any one aircraft, and mimic placement restrictions. If hazardous material is limited to one strip per aircraft, it should be possible to actually load these items near a door or off-load ramp as necessary. Since the PDRC was not provided with any regulations on how to load hazardous cargo, or if the different classes of hazardous cargo require special handling, this solution seemed reasonable.

The hazardous cargo strips are assigned to aircraft so as to maximize the remaining capacity of the aircraft. Thus, if a strip fits on many aircraft, it is assigned to the one whose resulting minimum capacity, as a percentage of its empty capacity, is maximum. For example, a strip has length 112", and weight 1 STON and it fits on two C-141's whose remaining length is 224" and 336", and whose allowable ACL is 2 and 4 STONS, respectively. If the strip were placed on the first aircraft the remaining length is 112", or 10% of a C-141; the remaining ACL is 1 STON or 3%. Similarly on the second aircraft the remaining length is 20% and the remaining ACL is 9%.

The minimum capacity on each plane is its ACL. Since the remaining ACL of the second aircraft with the strip placed on board is larger, the strip is assigned to the second aircraft. This procedure tries to keep the space on available aircraft as large as possible to accommodate the non-hazardous strips. Also note that the assignment of hazardous cargo to aircraft only matters when some of the aircraft are partially loaded.

The net effect of pre-loading the hazardous cargo is to partially load a number of aircraft equal to the number of hazardous strips plus those aircraft already partially loaded with hazardous material. Thus, aircraft with hazardous cargo are treated just like partially loaded aircraft when the algorithm continues to the second stage.

2.4.3 Multi-Constraint Multiple Knapsack Heuristic

At this stage in the execution of ALPE there exists a set of compatible strips and a set of aircraft capacities. The problem, now, is to combine the strips to generate a reasonable number of aircraft loads.

2.4.4 Multi-Constraint Knapsack Problem

A multi-constraint knapsack (MCK) problem is a binary integer program where the number of variables greatly exceeds the number of constraints, and all data are non-negative. The name "knapsack" problem arises from the similarity to a hiker's decision problem, where each item has an associated weight, volume, and utility. The hiker must decide to which items to take along subject to the volume of his pack and the amount of weight he is willing to carry. The mathematical formulation of this problem is

$\max \sum_j c_j x_j$

s.t.

$\sum_j v_j x_j \leq V$

$\sum_j w_j x_j \leq W$

$x_j \epsilon \{0, 1\}$

where all data are non-negative

If there were only one aircraft to load, and it was not required to lift all the cargo, then the problem facing that aircraft is a MCK. The problem that we must solve is one where there are many knapsacks, aircraft, and we wish to carry as much of the cargo as possible with the minimum number of aircraft. This problem will be called the multi-constraint multiple knapsack (MCMK) problem. A formulation of the MCMK problem is

$\max \sum_i \sum_j c_{ij} x_{ij}$

s.t.

$\sum_i x_{ij} = 1 \; for \; all \; j$

$\sum_j v_j x_{ij} \leq V_i \; for \; all \; i$

$\sum_j w_j x_{ij} \leq W_i \; for \; all \; i$

$x_{ij} \epsilon \{0, 1\}$

all data is non-negative

Note that the problem shows the staircase structure of a problem suitable for decomposition.

2.4.5 Model Development

The solution strategy common to the MCK problem must be generalized to fit the problem that LOADER must solve, a MCMK. Suppose with each aircraft there is an associated loadmaster. Each loadmaster possesses a list of all the strips that fit on his aircraft, and knows the cargo already loaded on his aircraft. Now based upon the strips that fit on his aircraft, each loadmaster decides which strip he would like to add to the current load. Since more than one loadmaster may desire the same strip, a way of breaking these conflicts must be decided upon.

When a loadmaster desires a strip, he must submit a bid to the chief loader. This bid must accurately reflect the utilization that the desired strip will add to the current load. The chief loader finds the maximum bid, and assigns the desired strip to that aircraft. The loadmasters then update their lists of feasible strips, and submit new bids. This proceeds until no strip will fit in any aircraft, or the strip list is exhausted.

2.4.6 Utility Functions

Many heuristics used to solve the MCK problem rely on designing appropriate utility functions. This utility is often the ratio of a profit function to some measure of resource consumption. The heuristic selects that feasible item whose profit to consumption ratio is largest among the items not yet selected. If all the strips will fit on the aircraft supplied, there is no competition for space on the aircraft. When this is true, it does not make sense to use a resource consumption factor in the utility function. Thus, the utility should be based on the profit function alone.

The profit function for the aircraft loading problem should reflect the utilization of the aircraft sortied and their number. One way to maximize aircraft utilization is to make the profit function depend on the reciprocal of a increasing function of the remaining aircraft capacity. Thus, as the aircraft capacity diminishes, the reciprocal increases. A function currently being implemented considered in ALPE is the minimum of the reciprocal of the remaining cargo bay length, and the reciprocal of the remaining ACL, both appropriately scaled.

When all the strips will not fit on the aircraft provided, either due to inherent infeasibility, or the solution generated by WIDTH-FIT, there

is competition for scarce resources. In this case a penalty, or resource consumption factor must be taken into account by the utility function. When considering resource consumption with respect to a candidate strip, three factors for each resource come into play. These factors are the total amount of the resource consumed so far after the candidate is added to the solution, the amount of resource remaining, and the future demand for the resource. The penalty function should be proportional to the first and third of these factors and inversely so to the second. An exposition on computational results for MCK with various utility functions can be found in Loulou and Michaelides [1979] and Gavish and Pirkul [1985].

LOADER uses a utility function to compute "bids". The concept of having a bid that each aircraft submits for the strip that it wishes allows great flexibility for ALPE. Since the utility function governs the assignment of strips to aircraft, changes in the utility function can greatly influence the solution that ALPE generates, and the changes are extremely easy to implement.

2.4.7 Implementation of MCMK

The implementation of the MCMK heuristic requires initialization of the utility function, finding the maximum utility, and then updating the utilities. Instead of computing the bids for each aircraft over all the items, LOADER computes the bids for each item over all the aircraft. This structure, a doubly nested loop iterating over aircraft for each item, allows the bid updates to be performed more efficiently.

An array stores the best bid for all strips, along with the aircraft to which the bid corresponds. If a strip does not fit on any aircraft, its bid is negative. As these strip bids are computed, the current maximum and second largest bid are also stored. So, after initialization the maximum bid and corresponding aircraft data are available. Otherwise, the maximum bid is negative so no strip fits on any aircraft.

The strip which achieves the maximum can be assigned to that aircraft, the strip marked as packed, and the aircraft capacities decremented. At this time, the second largest bid replaces the maximum bid, subject to being updated.

After a strip is assigned to an aircraft, the bids of all unpacked strips are compared to a tentative bid for the strip on that aircraft. If the tentative bid

exceeds the strip's bid, the bid is updated and compared to the maximum two bids. If a strip's bid was determined by its utility on the aircraft just updated, and that strip is no longer feasible, a new bid must be computed for that strip.

After the updates are finished, the process repeats until the maximum bid is negative. At this time, either all the strips are packed or the remaining strips fit on no aircraft. In the case that all cargo must be moved, the maximum bid being negative could trigger a call to a subroutine that would decide if an additional aircraft could be supplied. Computation could then proceed after re-initialization of the bids.

3 Issues

This section deals with the inherent limitations of the algorithm and options that should be empirically tested.

3.1 Partially Loaded Aircraft

When partially loaded aircraft are present, it is unclear if the algorithm should be allowed to reload these partially loaded aircraft. If the algorithm must reload these aircraft, then a cargo manifest must be passed to the algorithm along with each partially packed aircraft. Unpacking partially loaded aircraft will be detrimental to the running speed of the algorithm, but it might be necessary for satisfactory performance. For example, if two aircraft each 60% full arrive to find one item which requires 45% of their ACL, an additional aircraft would be needed, whereas re-packing the aircraft may allow the item to fit on the two aircraft.

It is assumed that aircraft will not be repacked; the given aircraft will be filled and unmoved cargo reported.

3.2 Compatible Aircraft

WIDTH-FIT generates plane width strips to pass to LOADER. The only circumstance when ALPE deals with different width aircraft is in handling outsize cargo. For this case, strips are generated for a C-5 and placed on board by the first call to LOADER. After this step the remaining cargo bay of each C-5 is considered to be two cargo bays half as wide. This should make the C-5 resemble a C-141 closely enough for estimation purposes. For civilian aircraft, and other less homogeneous aircraft types this may not work.

This restriction makes it difficult for ALPE to make loads for a diverse set of aircraft in one pass. As mentioned in previous PDRC monthly reports, ALPE expects to be given a fixed number of one type of aircraft, and C-5's, if needed. To remove this restriction, an additional procedure to assign cargo to aircraft types would be needed. This procedure would have to decide, based on the cargo list, and aircraft present, which cargos would be loaded on each plane type. After this decision has been made, WIDTH-FIT, and LOADER could proceed to generate aircraft loads. In

previous discussions with ORNL, a procedure of this type has been referred to as an aircraft-cargo matcher.

3.3 Width-Fit

WIDTH-FIT is an implementation of a Next-Fit Decreasing Height bin-packing heuristic. There are only two significant differences. The first is that once a strip is "opened", (an item is placed in it), all the remaining narrow items are examined to see if they fit, until the strip is full. A true NFDH heuristic would declare the strip full as soon as an item failed to fit.

The second difference is that weight limits must be checked. Since we wish that every strip can be loaded onto at least one of the aircraft on the list, the weight of a strip cannot exceed the maximum of the remaining ACL's. Difficulties may arise when many strips approach this upper bound. For example, if one aircraft has a large ACL, and the remaining aircraft are partially packed, WIDTH-FIT could generate strips whose weight is near that of the largest ACL. These strips would then fit only on that one aircraft, and not be feasible for the other aircraft.

This may seem disheartening. Recall, however, that the length of a strip is determined by the longest item assigned to it, and all the items in a strip are bottom-left justified. Thus, the linear density of the cargo, and the inherent inefficiencies of the NFDH heuristic bound the weight of any strip.

Also, if the maximum of the remaining ACL's was not used, but rather the average ACL or some other measure, the length of the strips as a whole would increase. It may be that this increase in total length would also make it impossible to pack all the strips.

While it is probably true that this modification (ignoring weight restrictions) of Coffman's NFDH heuristic does not affect its worst case algorithmic performance, its not yet clear whether the extra effort is worth the expected improvement in performance. Also, the threshold values for declaring a strip full can play an important part in making tradeoffs between algorithm speed and solution quality. Both of these questions and that of choosing the proper bound for the weight of a strip must be answered empirically.

3.4 Utility functions

The most challenging decision in the succesful implementation of ALPE is the proper choice of the utility function in LOADER. A myopic function to maximize aircraft utilization might be

$\max_i \min\{1/RL_i, 1/RACL_i\}$

where RL_i is the remaining length of the cargo bay after strip i is added,

and $RACL_i$ is the remaining ACL.

This function assigns the strip that increases the least used capacity the most to the aircraft in question. Note that the function computes the cost only for one aircraft. It could be computed for each aircraft, then the maximum taken, to assign strips to many aircraft.

The major drawback of this function is that by trying to increase the aircraft utilization as much as possible at each step it may get trapped into making bad choices. This is a pitfall common to many greedy heuristics. MCK heuristics overcome the myopia of the above function by factoring in to the utility function the future demand for the resource, the amount of the resource consumed so far, and the amount of resource remaining.

The MCMK for the problem that LOADER must solve possesses two constraints for each aircraft. If the MCK solution methodology is used, then each aircraft capacity constraint would be considered a resource. It may be possible to consider the total amount of cargo bay length and ACL present as the resources of the problem. This aggregation would decrease the number of constraints from twice the number of aircraft to two. The decrease in the number of constraints would speed up computation of the utilities.

The final choice of utility function must be decided by empirical testing. Regardless of the arguments made on paper for certain functions, their impact on the solution cannot be adequately predicted by theory alone.

8-24-645

WORK PLAN

for the

Enhancement and Validation of ADANS Airlift Load Planning Algorithms

Submitted to:

Martin Marietta Energy Systems, Inc.
Post Office Box M
Oak Ridge, TN 37831

Submitted by:

School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

July 19, 1988

The following tasks represent the work plan for the development and implementation of the ADANS Airlift Load Planning Algorithm. The plan recognizes work already accomplished today as well as proposed effort to complete each task.

The major difference between the proposed work plan and the proposed effort are (1) the requirement for implementation on the DFCS has been dropped because of the time constraints on the current effort, and (2) artificial intelligence modelling and analysis activities were removed from the work plan.

The following pages describe each task of the work plan, associated deliverables, and due dates.

Task 1: Enhancement of the ADANS Airlift Load Planning Algorithm

An enhanced, working prototype algorithm will be made operational on the IBM 9375 minicomputer at Georgia Tech's Production and Distribution Research Center.

The improved airlift load planning algorithm will be able to recognize cargo loading restrictions with respect to height, weight, width, volume, cube, lineal feet, footprint, and preassigned priority, and to identify incompatible or hazardous cargo mixes.

This enhanced load planning algorithm will also be able to divide loads while maintaining the integrity of large pieces of equipment, such as wheeled vehicles. For example, the algorithm should avoid allocating portions of a vehicle to separate aircraft. These loading limitations will be defined with respect to each of the forty different types of aircraft, both military and civilian, currently being used by MAC. The algorithm will also handle passenger loads, again by specific aircraft type. Subject to applicable aircraft capacity constraints the algorithm will optimize aircraft-to-cargo matching.

Based on previous Georgia Tech research for ORNL, the methodology with the most potential for accommodating level4 data is the class of "fitting" (packing) algorithms. The published research in this area deals primarily with one-dimensional problems (i.e., those with a single measure of "size" of items to be loaded). The current research effort will focus on extending models in this class to account for level4 data in the development of the proposed algorithm.

The recommended algorithm will report the number of aircraft, by type, used to move the cargo, and also report partially loaded aircraft and any portion of a movement requirement not airlifted.

Deliverable: A report describing the conceptual and mathematical aspects of the approach to be followed in adapting the ADANS load planning algorithm to realistically address the variety of MAC aircraft to cargo matching situations.

Due Date: September 30, 1988.

Task 2: Validation of the Load Planning Algorithm Against Realistic Data

Emphasis in algorithm validation will be upon both the accuracy and the computational cost (i.e. runtime) involved in running the load planning routine, and on the trade-off between these two competing objectives. The issue of how best to validate such an algorithm is a difficult one to address, and rests ultimately with use of the approach in practice.

Tests of robustness will be applied to evaluate the consistency of results with regard to variations in (1) the presentation of the data (i.e., the ordering of the cargo) and (2) the size of the total cargo to be moved.

Tests will also be performed to compare the ADANS loading algorithm against other similar algorithms. The algorithm will be tested against the current preliminary approach used by Distinct. For this test a sample of 100-150 requirements will be generated at level2 for run application of the Distinct algorithm. ORNL will apply the Distinct algorithm to this set of data and obtain results for Georgia Tech to make the subsequent comparisons. Georgia Tech will also explore the possibility of testing against the procedures developed for MAC by the Air Force Academy.

This will provide indications of the worth of models incorporating multiple measures and special constraints, as in the proposed approach. The algorithm will also be validated against procedures oriented towards handling the details of loading (or "packing") single aircraft (e.g. the CALM loading routine). Comparison of the ADANS algorithm against the CALM loading algorithm will be performed by Tech. (A "C" version of the CALM algorithm has recently been made available to Georgia Tech for another purpose.)

Deliverable: A written proposal and justification for the validation tests to be performed.

Due Date: October 15, 1988.

Task 3: Working Demonstration of the Prototype Algorithm

Georgia Tech will provide a working demonstration of the enhanced load planning prototype algorithm on the DFCS. Details of the demonstration will be worked out between Tech and ORNL. The objective is to show the degree of accuracy and speed of computation associated with algorithm application to a variety of typical MAC movement requirements. For the purposes of definition, "demonstration" refers here to Georgia Tech making available a copy of the enhanced prototype load planning algorithm, its user implementation procedure, input data and results of tests.

The demonstration prototype will be designed, developed, and implemented so as to accommodate absorption into the Distinct scheduling algorithm without major redesign and recoding.

Deliverable: A working demonstration copy of the enhanced load planning prototype algorithm.

Due Date: November 30, 1988.

Task 4: Algorithm Documentation, Test Results and Code

Georgia Tech will write a technical user's manual, documenting the enhanced load planning algorithm's mathematical structure, its user interface, and a summary of the results of testing it against realistic data. Tech will also make available to ORNL at this time the algorithm's source and executable codes.

Deliverable: A technical user's manual for the prototype Algorithm.

Due Date: December 31, 1988.

PDRC Report Series 89-01
May 1989

AIRCRAFT LOAD PLANNING ESTIMATOR FINAL REPORT

by
Christopher Hane
John J. Jarvis
H. Donald Ratliff
Bryan R. Stutzman

Georgia Institute of Technology
School of Industrial and Systems Engineering
Atlanta, GA 30332

Contents

List of Tables

List of Figures

1 Introduction

This report presents an algorithm, ALPE (Aircraft Load Planning Estimator), for determining the number of aircraft required to lift a given set of cargo; its implementation, and computational results. This algorithm uses Level 4 data as input, specifically, length, width, height, weight, priority and cargo category codes present in SRF and TUCHA records. The algorithm reports the number of each plane type used, the amount of usable space remaining on each of the aircraft, and any cargo that was not moved.

Many areas of improvement over current load estimation algorithms are implemented in this algorithm. Cargo integrity is maintained through every step of the algorithm, i.e. no item is assigned to more than one aircraft. Hazardous cargo items are also identified and handled separately. Also, optimization techniques play a role in assigning cargo to aircraft.

Based on previous research for ORNL, the methodology used to implement this algorithm is a combination of bin-packing and multi-constraint knapsack algorithms. The structure is three part: pre-processing, assigning individual items to strips one plane width wide, and combining strips to form aircraft loads. The assignment of items to strips is a bin-packing problem. The subroutine to perform this algorithm is called WIDFIT. Combining strips to form aircraft loads can be performed by a multi-constraint knapsack heuristic, it will be referred to as LOADER. There are actually two versions of LOADER within the ALPE code, each performs under separate option codes.

1.1 Model

A convenient model of the algorithm for packing one type of empty plane is the following. Initially, we have an unbounded rectangle whose width is that of the cargo bay. We wish to place items in this rectangle so that the length of the rectangle containing all of the items is minimized. However, we also wish that the packing is decomposable into units that do not exceed the ACL of the plane type or its cargo bay length. The decomposability can be achieved by a level-oriented packing heuristic with a side constraint on weight (WIDFIT). Once we have assigned cargo pieces to these units, called strips, the strips must be collected together to form good plane loads.

Since all strips are one plane width wide, the only constraints for an

aircraft load are the overall length and ACL. To assign strips to one empty plane, we could solve a binary integer program with a variable for each strip, and two constraints, for length and weight. The objective of the program is to maximize the utilization of the aircraft, or minimize slack in the constraints. A similar type of problem is called a multi- constraint knapsack problem. Its formulation is discussed in section 2.4.6. There are fast heuristic procedures to solve this problem.[1] ALPE builds upon such a heuristic to handle multiple aircraft, partially packed aircraft and incompatible cargo (LOADER).

In general, the presence of hazardous cargo, outsize cargo and partially packed aircraft complicate the actual implementation of the above ideas. However, even with these complications the framework is robust enough to handle them. When outsize and oversize cargo are present, strips must be generated for the outsize cargo using a C-5 bay width, and for the other cargo using the width of the available plane type. If C-5's are available, and the outsize cargo does not fill the C-5, two regular width strips can be combined to form a C-5 strip.

2 General Algorithm Structure

The initial stage of the ALPE algorithm is a pre-processing one. The individual handling requirements of the diverse types of cargo require that the cargo list be separated. Hazardous and outsize cargo are two examples of cargo classes that require special attention. In general, cargo is separated into classes by its hazardous tag and its size tag. In addition to separation by classes, the pre-processing stage also sorts the cargos in the classes.

The WIDFIT subroutine generates strips of cargo whose width is no more than the width of the cargo bay. This subroutine is executed for each of the non-bulk cargo classes. Bulk cargo is assumed to be palletized, and therefore either one or two pallets form a strip. As the individual cargo classes require different loading techniques, their strips do also.

The LOADER subroutine combines strips to form aircraft loads. Thus it must be executed for each different width of strips generated by WIDFIT, and each hazardous cargo class present. If all planes were C-141's or C-130's

[1]Loulou and Michaelides, New Greedy-like Heuristics for the Multidimensional 0-1 Knapsack Problem, Operations Research, vol. 37, no. 6, 1979.

with one class of cargo only one pass of LOADER is necessary. However if outsize items along with other cargo are present, two passes through LOADER must be executed.

2.1 Inputs to ALPE

There are two input lists that ALPE uses. The first is the cargo list. This list must include the information present in the SRF Force Cargo Detail Record. Among the data in this record, the important fields are the cargo category code, cargo dimensions, weight, and number of pieces. Optionally, priorities may be assigned to each item.

The second list provided to ALPE is the available aircraft list. This list must specify the aircraft type, including configuration if necessary, aircraft capacities, and number of aircraft that share these properties. It is allowable that the aircraft on the list be partially loaded. In this case the aircraft capacities are the remaining capacities, and there must be a flag indicating whether hazardous cargo is on board. An option code allows to specify all available aircraft as empty, thus avoiding the specification of individual aircraft capacities.

Also, needed by ALPE is a file of solution options. These option codes govern the choice of functions used in computing the solution to the current problem. The choice of options is discussed in the implementation section and their impact in the computational results section.

2.2 Pre-Processing

Due to the variety of item sizes and possible incompatible cargo types, pre-processing the data is necessary for guaranteeing good performance in WIDFIT. Theoretical results for bin-packing algorithms indicate that sorting the data can have a profound affect on algorithmic efficiency.[2] The discrepancies of cargo types: outsize, bulk, hazardous, etc., require individual treatment for loading. Therefore dimensional data for these types are collected separately.

[2]Coffman, E.G., et al., Performance Bounds for Level-Oriented Two- Dimensional Packing Algorithms, SIAM J. of Comput., vol. 9, no. 4, Nov. 1980, pp. 808-826.

2.2.1 Hazardous Cargo

Hazardous cargo, those pieces whose first position cargo category code is either D, E, K or L (see table T-18 in Appendix T of MAC pamphlet 76-2), require special placement on board an aircraft. To mimic this placement restriction, all dimensions of hazardous items are stored in an array separate from the other data. After the hazardous items have been placed in plane width strips by WIDFIT, each strip is assigned to a plane, whose other cargo is compatible with this strip. This implies that cargo that share a hazardous tag must be compatible. Also at least one call to LOADER is made for each hazard class.

2.2.2 Outsize Cargo

Another class of cargo that requires separate handling is outsize cargo. Outsize cargo is defined as any air transportable cargo which requires the use of a C-5 aircraft. Since outsize cargo requires a specific plane type, all dimensional data for outsize cargo is stored in a separate array.

2.2.3 General Cargo

To increase the efficiency of the bin-packing subroutine, the data must be ordered by non-increasing length. WIDFIT is similar to Coffman, et al's Next-Fit Decreasing Height (NFDH) heuristic. Since ALPE performs packings on a horizontal surface (the cargo bay), the dimensions of concern are length and width, where length corresponds to height in the literature.

In the NFDH heuristic rectangles are packed left-justified on a level until there is no more room to the right to allow the next rectangle to fit. When this occurs, a new level is defined, the old level is closed, and packing continues on the new level. If item length is strongly correlated with its width, then sorting by decreasing length, also tends to order the items by width. This causes problems for the NFDH heuristic. An example of a NFDH and a WIDTH-FIT packing is shown in Figure 1.

For example, if many of the items have width exceeding one- half the width of the cargo bay and these items are adjacent in the input list, NFDH will pack only one item per level. In this situation it would be advantageous to examine more than just the next item on the list to try to find an item that will fit next to the wide item.

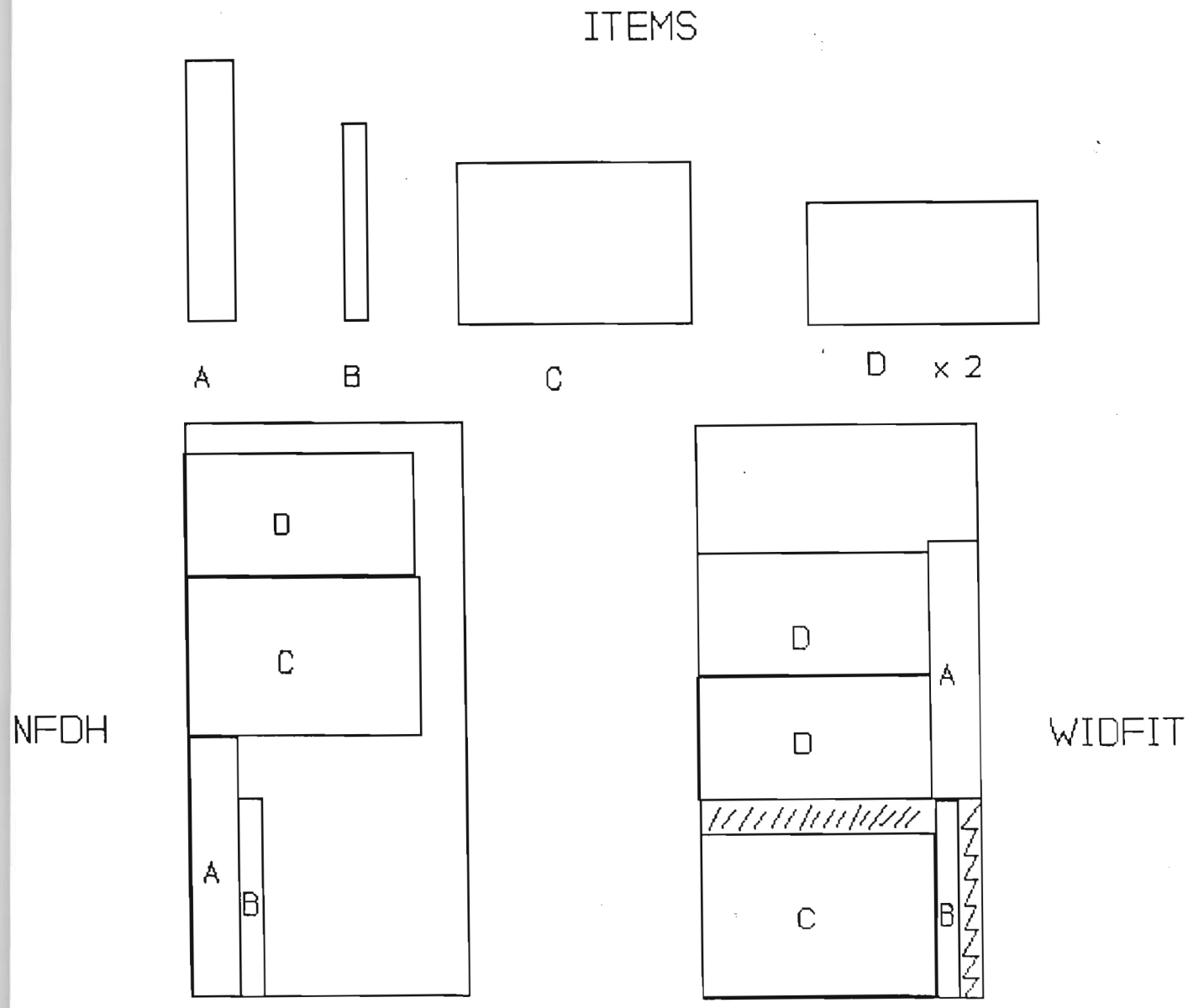

Figure 1: NFDH

This argument is the motivation for separating the cargo into lists of items that exceed one-half the width of the cargo bay and the narrower items. Thus, all compatible cargo are partitioned into wide and narrow lists. Then when the items are being assigned to strips, a wide item is placed on the strip then the narrow list is searched for items that will fit adjacent to it. How this is actually performed will be discussed in the next section.

The output of the pre-processing stage is a list of wide items and a list of narrow items for each compatible non-bulk cargo class. The bulk cargo classes are assumed to be palletized. All cargo class lists are ordered by non-increasing length.

If the items in Figure 1 are all members of the same cargo class, then items C and D are in the wide list, with A and B in the narrow list. The WIDTH-FIT packing in Figure 1 shows two strips, the strips contain items C and B, and items D, D and A. The NFDH packing only packs one of the D items. Also, item A does not fit alongside item C.

2.3 WIDFIT

2.3.1 Introduction

This stage of the ALPE system assigns each item to a strip, a rectangle the width of the aircraft whose length is determined by the longest item assigned to the strip. This subroutine is executed once for each of the cargo classes described in the previous section. It is necessary to separate the strips generated for each cargo class because the strips in the hazardous classes require special attention, and those in the outsize classes require a C-5 aircraft. This section will describe how WIDFIT assigns items to strips.

2.3.2 Making Strips

The main body of WIDFIT consists of two doubly-nested loops. The first loop iterates over the items in the wide item list. Only one wide item can fit on a strip, so each wide item is placed on its own strip. The inner loop iterates over the narrow item list.

Once a wide item is placed on a strip, the unassigned narrow items are checked to see if they fit in the remaining width and weight of the strip. If

a narrow item is placed on a strip, the remaining width and weight of the strip are updated, and the item is marked as packed. If the length of the narrow item exceeds that of the wide item it may be possible to place an additional wide item on that strip.

This extra wide item distinguishes WIDFIT from a level- oriented bin-packing algorithm; it is a guillotine packing algorithm. The term guillotine arises from the cutting stock literature, where the cuts in a sheet must be perpindicular to a side and cross the entire sheet. Thus our strip is a sheet with variable length and width equal to a cargo bay.

After all the wide items are assigned to strips, there are two possible situations to handle. The first is that there are no unpacked narrow items. In this case, WIDFIT can terminate because all items are assigned to strips.

The second case is that there are some unpacked narrow items remaining. When this happens the second of the doubly-nested loops is entered. The procedure here is the same as that in the first double loop except that both loops iterate over the remaining narrow items. Thus, a narrow item is assigned to an empty strip, then the remaining unpacked narrow items are checked to see if they fit adjacent to that item on the strip.

2.3.3 Output

The output of WIDFIT is the list of strips with their dimensions and a manifest of the items on that strip. Also, any item whose length or weight exceeds the maximum aircraft capacity of that dimension can immediately be classified as not movable. This is true since partially packed aircraft are not repacked.

Again, in Figure 1 there are three strips. Strip 1 has length equal to that of item 1 and weight equal to the sum of the weights of items 1 and 4. Similarly, strip 2 has length equal to that of item 2 and weight equal to that of items 2, 5 and 6. The same holds for the third strip consisting of items 3 and 7.

2.4 LOADER

2.4.1 Introduction

The purpose of LOADER is to take all the strips generated by the calls to WIDFIT, and combine them to form good loads. LOADER must be called once for each outsize and non-outsize hazard cargo class.

2.4.2 Multi-Constraint Multiple Knapsack Heuristic

At this stage in the execution of ALPE there exists a set of compatible strips and a set of aircraft capacities. The problem, now, is to combine the strips to generate a reasonable number of aircraft loads.

2.4.3 Multi-constraint Knapsack Problem

A multi-constraint knapsack (MCK) problem is a binary integer program where the number of variables greatly exceeds the number of constraints, and all data are non-negative. The name "knapsack" problem arises from the similarity to a hiker's decision problem, where each item has an associated weight, volume, and utility. The hiker must decide to which items to take along subject to the volume of his pack and the amount of weight he is willing to carry. The mathematical formulation of this problem is shown in Figure 2.

$$\begin{gathered}
\max \textstyle\sum_j C_j X_j \\
s.t. \\
\textstyle\sum_j V_j X_j \leq \mathrm{V} \\
\textstyle\sum_j W_j X_j \leq \mathrm{W} \\
X_j \epsilon \{0,1\}
\end{gathered}$$

where all data are non-negative

Figure 2: Multi-constraint Knapsack Problem

If there were only one aircraft to load, and it was not required to lift all the cargo, then the problem facing that aircraft is a MCK. The problem that we must solve is one where there are many knapsacks, aircraft, and we

wish to carry as much of the cargo as possible with the minimum number of aircraft. This problem will be called the multi-constraint multiple knapsack (MCMK) problem. A formulation of the MCMK problem is shown in Figure 3.

$$\max \sum_i \sum_j C_{ij} X_{ij}$$
$$s.t.$$
$$\sum_i X_{ij} = 1 \quad for\ all\ j$$
$$\sum_j V_j X_{ij} \leq V_i \quad for\ all\ i$$
$$\sum_j W_j X_{ij} \leq W_i \quad for\ all\ i$$
$$X_{ij} \epsilon \{0, 1\}$$
all data is non-negative

Figure 3: A MCMK Problem

Note that the problem shows the staircase structure of a problem suitable for decomposition.

2.4.4 Model Development

The solution strategy common to the MCK problem must be generalized to fit the problem that LOADER must solve, a MCMK. Suppose with each aircraft there is an associated loadmaster. Each loadmaster possesses a list of all the strips that fit on his aircraft, and knows the cargo already loaded on his aircraft. Now based upon the strips that fit on his aircraft, each loadmaster decides which strip he would like to add to the current load. Since more than one loadmaster may desire the same strip, a way of breaking these conflicts must be decided upon.

When a loadmaster desires a strip, he must submit a bid to the chief loader. This bid must accurately reflect the utilization that the desired strip will add to the current load. The chief loader finds the maximum bid, and assigns the desired strip to that aircraft. The loadmasters then update their lists of feasible strips, and submit new bids. This proceeds until no strip will fit in any aircraft, or the strip list is exhausted.

2.4.5 Utility Functions

Many heuristics used to solve the MCK problem rely on designing appropriate utility functions. This utility is often the ratio of a profit function to some measure of resource consumption. The heuristic selects that feasible item whose profit to consumption ratio is largest among the items not yet selected.

The profit function for the aircraft loading problem should reflect the utilization of the aircraft sortied and their number. One way to maximize aircraft utilization is to make the profit function depend on the reciprocal of a increasing function of the remaining aircraft capacity. Thus, as the aircraft capacity diminishes, the reciprocal increases. These functions are described in section 3.

When all the strips will not fit on the aircraft provided, either due to inherent infeasibility, or the solution generated by WIDFIT, there is competition for scarce resources. In this case a penalty, or resource consumption factor must be taken into account by the utility function. When considering resource consumption with respect to a candidate strip, three factors for each resource come into play. These factors are the total amount of the resource consumed so far after the candidate is added to the solution, the amount of resource remaining, and the future demand for the resource. The penalty function should be proportional to the first and third of these factors and inversely so to the second. An exposition on computational results for MCK with various utility functions can be found in Loulou and Michaelides [1979] and Gavish and Pirkul [1985].

LOADER uses a utility function to compute "bids". The concept of having a bid that each aircraft submits for the strip that it wishes allows great flexibility for ALPE. Since the utility function governs the assignment of strips to aircraft, changes in the utility function can greatly influence the solution that ALPE generates, and the changes are extremely easy to implement.

2.4.6 Implementation of MCMK

The implementation of the MCMK heuristic requires initialization of the utility function, finding the maximum utility, and then updating the utilities. Instead of computing the bids for each aircraft over all the items,

LOADER computes the bids for each item over all the aircraft. This structure, a doubly nested loop iterating over aircraft for each item, allows the bid updates to be performed more efficiently.

An array stores the best bid for all strips, along with the aircraft to which the bid corresponds. If a strip does not fit on any aircraft, its bid is negative. As these strip bids are computed, the current maximum and second largest bid are also stored. So, after initialization the maximum bid and corresponding aircraft data are available. Otherwise, the maximum bid is negative so no strip fits on any aircraft.

The strip which achieves the maximum can be assigned to that aircraft, the strip marked as packed, and the aircraft capacities decremented. At this time, the second largest bid replaces the maximum bid, subject to being updated.

After a strip is assigned to an aircraft, the bids of all unpacked strips are compared to a tentative bid for the strip on that aircraft. If the tentative bid exceeds the strip's bid, the bid is updated and compared to the maximum two bids. If a strip's bid was determined by its utility on the aircraft just updated, and that strip is no longer feasible, a new bid must be computed for that strip.

After the updates are finished, the process repeats until the maximum bid is negative. At this time, either all the strips are packed or the remaining strips fit on no aircraft. In the case that all cargo must be moved, the maximum bid being negative could trigger a call to a subroutine that would decide if an additional aircraft could be supplied. Computation could then proceed after re-initialization of the bids.

2.5 ADDAIR

ADDAIR opens empty aircraft for loading. The number of aircraft opened depends on the values of the parameters NUMFIX, and IADD. This subroutine only affects how empty aircraft are opened ofr the algorithm, partially packed aircraft are always open initially.

If NUMFIX is set to zero, the initial number of aircraft opened and all subsequent openings are made according to the LP bound based on the remaining amount of cargo, or the number of remaining aircraft. If NUMFIX is set to 1, the number of aircraft opened is determined by IADD, e.g. if IADD = 5, then aircraft are added in groups of five.

The impact of these parameters can be seen in tables 35 and 36, where the short aircraft lists decrease the run times by a factor of more than 20 in some cases.

3 Computational Results and Analysis

The most challenging decision in the succesful implementation of ALPE is the proper choice of the utility functions in LOADER. The utility functions assign the cost and benefit to the placement of a strip on an aircraft. The cost to benefit ratio establishes the profit of that assignment. The tenative assignment with the maximum profit is the assignment implemented.

The profit or benefit of assigning an item to an aircraft is difficult to determine. When all items must be lifted, the amount of aircraft resources that will be consumed is known. It is equal to the amount of cargo on hand. However, the discrete nature of the items forces inefficiencies in loading; the challenge is to minimize these inefficiencies.

The loading problem differs from other applications of cost- benefit ratios. Here, the cost and benefit are both determined by the consumption of resources. The trick is to force the benefit to mimic "good" choices of consumption, and cost, a "bad" choice.

3.1 Cost functions

A cost function to maximize aircraft utilization would be similar to (1).

(1) cost $(i,j) = max\{IL_i/AL_j, IWT_i/RACL_j\}$
where IL_i is the i^{th} strip length, IWT_i is the i^{th} strip weight,
AL_j is the remaining length of the j^{th} cargo bay,
and $RACL_j$ is the remaining ACL of the j^{th} aircraft .

This function assigns the cost equal to the greatest percent of resource consumed of strip i on aircraft j. Function (1) penalizes the placement of an item where it would use a large amount of a scarce resource.

A second cost function, (2), has the same form as (1) except the denominators are replaced by the total amount of length remaining in the aircraft fleet, and the total amount of ACL remaining in the fleet. This function

seeks to avoid penalizing placement on any one aircraft, but assesses penalties for using the resources of the fleet as a whole. This function also need only be computed after the capacity of the aircraft fleet changes, not for each aircraft as with function (1).

The reasoning for this is function (1) applies the maximum penalty for exactly filling the cargo bay. This trait is not always desirable, however, it prevents an aircraft from filling up on one measure too quickly. Function (2) prevents utilization of the fleets assets (length and ACL) too quickly.

As a control function the other cost function used in the test cases was to assign all placements an equal cost.

The computational results indicate that with long aircraft lists, function (2) requires too much time without a compensory increase in aircraft utilization. Those occasions where assigning no true cost to placements achieved the minimum number of aircraft, appear to be more related to the fact that aircraft are made available only one at a time. This method of forcing the algorithm to place a strip on the aircraft, if one fits, is a good way to get a fair solution rather quickly. It need not be applied with the uniform cost function.

When shorter aircraft lists were used, no difference was found in solution quality. Short aircraft lists, verses long lists are explained in section 3.5. Hence, limiting the size of the aircraft lists is a good way to reduce run times, without harming solution quality.

3.2 Profit functions

The profit function determines the benefit of placing an item on an aircraft. The profit functions used both encourage the use of the resource which is more plentiful. They differ in the way this is achieved.

The first profit function is the opposite of the first cost function.

(1) $\text{profit}(i,j) = \min\{IL_i/AL_j, IWT_i/RACL_j\}$

where IL_i is the i^{th} strip length, IWT_i is the i^{th} strip weight,

AL_j is the remaining length of the j^{th} cargo bay,

and $RACL_j$ is the remaining ACL of the j^{th} aircraft.

This function, in conjunction with the first cost function, rewards the balanced use of the aircraft's resources, i.e. the largest profit to cost ratio

will be for an item that uses equal percentages of both resources. Without a cost function, this profit function rewards the placement of that strip whose smaller consumption is largest.

The second profit function determines which resource on the current aircraft is most plentiful and rewards its use. This function strives to make alternating placements of items with large weights and large lengths. Thus it, too, rewards the balanced use of the aircraft's resources but it uses the resource information more directly.

This function may not perform well without a true cost function. Without a cost function this profit function does not necessarily make alternating placements. This results because the item that uses the most of the plentiful resource may also use a lot of the more scarce resource. If no penalties apply for using the scarce resource this profit function may appear greedy-like in its selection.

The computational results indicate that profit function 2 has a slight advantage over the first profit function. It frequently finds the minimum number of C-5 aircraft, this then causes some repercussions for the number of C-141 aircraft. It also usually leaves more unused capacity, than the first profit function.

3.3 Priority functions

The computational results did not favor one priority function over another.

3.4 Empty Aircraft Penalties

If no additional penalty is assigned to the first item placed on an empty aircraft, many unnecessary aircraft may be used. In data set 2, prior to the implementation of the extra cost, the number of C-141's used ranged from 0 to 18. After doubling the cost of the initial item placed on a strip, no set of option codes used a C-141. A greater penalty may force one aircraft at a time to be loaded. However this approach is better suited to the proper setting of NUMFIX and IADD, to gain the better run time.

3.5 Length of Aircraft Lists

Short aircraft lists implies a small fixed number of aircraft were available at any one time. The original test cases always had the lower bound number of aircraft available. This bound determined the length of the loops in LOADER. For short lists, instead of making all those aircraft available at once, only five at a time were available, i.e. IADD = 5. Earlier reasoning lead one to think that a restriction on the number of available aircraft would limit the freedom of the algorithm for placing strips. Glancing through the solution files, however, showed that the cases where strips were not placed on the same aircraft as the previous strip were rare, except of course when an aircraft was full. With this evidence in hand, the ALPE code was modified to allow the specification of long or short lists and the length of the lists. The list length of five was arbitrarily decided upon as long enough for flexibility, yet short enough to enhance run times.

Computational results indicate setting NUFIX = 1, and IADD = $\{3, \ldots, 10\}$ is adequate for all data sets. The extra flexibility given by setting NUMFIX = 0 is not needed and results in extremely long run times.

3.6 WIDFIT

WIDFIT is a modification of a Next-Fit Decreasing Height bin-packing heuristic. There are three significant differences. The first is that once a strip is "opened", (an item is placed in it), all the remaining narrow items are examined to see if they fit, until the strip is full. A true NFDH heuristic would declare the strip full as soon as an item failed to fit.

The second difference is that weight limits must be checked. Since we wish that every strip can be loaded onto at least one of the aircraft on the list, the weight of a strip cannot exceed the maximum of the remaining ACL's. Difficulties may arise when many strips approach this upper bound. For example, if one aircraft has a large ACL, and the remaining aircraft are partially packed, WIDFIT could generate strips whose weight is near that of the largest ACL. These strips would then fit only on that one aircraft, and not be feasible for the other aircraft.

The third difference is allowing extra wide items to be placed on a strip when the narrow item's length greatly exceeds that of the wide item. Note that no search is performed to find this extra wide item, it must come from

the same Level-4 record. This extra modification was necessary because the wide and narrow items are sorted separately.

While it is probably true that these modification (ignoring weight restrictions) of Coffman's NFDH heuristic does not affect its worst case algorithmic performance, its not yet clear whether the extra effort is worth the expected improvement in performance. This question question and that of choosing the proper bound for the weight of a strip must be answered empirically.

The use of the minimum ACL value for the upper bound on strip weight is not recommended. This value is often small enough to prevent WIDFIT from placing any items together. The merits of the average ACL value verses a non-binding bound have not been tested because all aircraft used in this study were empty.

Another important parameter in WIDFIT is the cargo bay width. This parameter is updated when C-5 cargo bays are split. The assumed cargo bay width is then one-half of the C-5's width, 228". This leads to strips of 114" being generated, a waste of 9" per strip on a C-141. If no outsize cargo are present, this problem does not occur.

This nine inch difference may not appear to be significant, but it is. The efficiency of the bin-packing depends on having many narrow items to fill space alongside the wide items. In the entire 8539 records of data less than 7.8% had widths less than 58", and 11% had widths less than 62". Thus more than 3% of the data set becomes ineligible for side by side placement when the cargo bay is decremented from 123" to 114".

If the cargo bay is not decremented better use of the narrow-bodied aircraft is possible at the expense of possible infeasibility on the C-5.

3.7 LP Relaxation Bounds

The LP relaxation bound for a single aircraft type is the maximum ratio test. For data sets where bulk cargo is present, the number of C-141 aircraft used is almost always within 5% of this bound. This good performance is related to the large numbers of pallets present in these data sets. The standard pallet is 108" long and the C-141 cargo bay is 1120" long. Thus, ten pallets fit in the bay, using 96.4% of its length. Alternately, the length bound overestimates the number of C-141's by 3.7%. With a large number of pallets, any small inefficiency in oversized loading can be washed out by

this good fit.

Without bulk items, ALPE still performs well. In only one data set, number nine, does an ALPE solution for C-141's exceed the LP bound by more than 5%. In data set nine, all but one solution is within 10% of the LP bound. The one exception is a solution 19% greater than the LP bound.

With C-5 aircraft the percentage is generally larger than 10%. This is due to the small number of aircraft, and the varying sizes and weights of outsize items. With a small number of aircraft to be loaded, one poor placement choice can cause an inefficient packing that cannot be masked by numbers alone. In the data sets with bulk items, where outsize items are a small fraction of the total items, every data set but one used at most 2 C-5's more than the LP bound. The data set which fell outside this range was number 2. Here, ALPE used as many as 19 additional C-5's. This is easily explained. The data contained an outsize record with 42 tanks, weighing 51 tons each. Using an empty ACL of 101 tons, the weight test predicts 1.98 tanks per aircraft. However, only one tank can fit on an aircraft so at least 42 aircraft are required, not the 24 predicted by the LP bound.

Among those data sets without bulk cargo, in two cases ALPE used the LP bound number of C-5's, twice it exceeded the bound by 2 planes or less, (7/5, 5/4), and once by 5 aircraft (18/13).

3.8 Pallets

The ALPE parameter for cargo bay length of a C-141 has been set a 1120", in accordance with MAC Pamphlet 76-2, attachment 3, page 33. Pallet dimensions are 108"l×88"w. These measurements restrict ALPE to packing a maximum of only ten pallets per aircraft. However the same reference above states that 13 pallets can be loaded on a C-141. These facts appear contradictory.

3.9 Additional Aircraft Types

Currently, ALPE is to use C-5 and C-141 aircraft. This can be easily modified, especially in the presence of an aircraft- cargo matcher.

The parameters in ALPE for empty aircraft dimensions, and wide item could be replaced with a common array, initialized by a data statement. In DISTINCT's Deliberate Load Planning Algorithm, the aircarft-cargo

matcher, and hierarchical cargo loading allow ALPE to be streamlined to handle one aircraft type and cargo class at a time. Entire subroutines, such as ADDBLK and ADDC5, could be discarded.

ALPE assumes C-5's and C-141's to load outsize and bulk cargo simultaneously on those planes. The need to split C-5 cargo bays arose from deciding which items should be placed on the remaining area of the C-5 and which on the C-141. When loading is hierarchical, this decision is made at an earlier stage. Thus when only one aircraft type is present, a slightly modified ALPE can handle any aircraft for which it has the data.

3.10 Explanation of Tables

Most data sets results are summarized in three tables in appendix 7. The first of these is the table describing the number of aircraft utilized by the algorithm for each combination of option codes. All results are for lifting all the cargo. The minimums are italicized, and maximums in bold, where necessary for emphasis.

The second table shows the run times for each set of option codes. All run times are wall clock time, not CPU seconds, reported for packing all the cargo with long aircraft lists, unless otherwise indicated. The actual length of the aircraft lists preceeds the third table, where it also denotes the number of aircraft used to pack some of the cargo.

For the third table of data sets one to eight, the expected percentage of lifted is the ratio of the number of aircraft available to the minimum number of aircraft reported in table one, i.e. the minimum upper bound. The actual amount of cargo lifted is the minimum, with respect to square feet and weight, of lifted cargo. For example, in data set one, cost=2, priority=1, and profit=2, theexpected percent of outsize cargo moved is 5/6. The actual amount moved is 92% by weight and 74% by square feet. Thus the entry in table 1.3 is .74/.833 = .888. Similarly, for oversize and bulk cargo the expected percent is 99/103. All oversize material is lifted, and for bulk 94% by weight, and 92% by square feet is lifted. So the entry is .92/.96 = .957.

If all the cargo is moved, the entry is a one. If more than the expected amount is moved the entry is an asterisk. For data set four all option combinations moved more than the expected amount or all the cargo.

4 IMPLEMENTATION

4.1 Introduction

This chapter is a detailed description of the implementation of the Aircraft Load Planning Estimator, ALPE. Each section of this report describes a program unit of ALPE. This description includes the purpose of the subroutine, its arguments, an explanation of its execution, and comments about choices made that could affect solution quality. Appendix 6 contains a dictionary of array and variable names.

Following this code description is a section on how to run this version of ALPE.

4.2 FORTRAN Code Description

All the code written for ALPE is FORTRAN 77. The include directive is no longer used.

4.2.1 MAIN Program

Purpose:

The main program controls the general execution of ALPE. It first calls the subroutines that read the data, (GETIT and GETPL), and then sorts, by non-increasing length, the items in each cargo class, (SORTER). Once this initialization is complete, the plane width strips are generated, (WIDFIT), aircraft loads determined (LLOAD or GLOAD), and necessary intermediate steps are performed (ADDC5 and ADDBLK). In the following description the routine LOADER will be referred to when either LLOAD or GLOAD is actually called.

Execution:

After the initialization by GETIT, GETPL, and SORTER, the packing subroutines, WIDFIT and LOADER, are executed for each aircraft class[3] present in the aircraft list. The outsize cargo is processed first. This is

[3]An aircraft class is a set of aircraft that share a similar cargo bay width, and cargo compatibility. Cargo compatibility is determined by the hazard marker for the aircraft, and outsize restriction. Thus a C-141 and C-130 are in a class if they contain the same type of cargo; a C-5 is never in the same class as a C-141.

performed so that the remaining cargo bay area can be partitioned into two dummy cargo bays for the loading of oversize and bulk cargo. The partitioning of a C-5 cargo bay into two narrow bays is performed by the subroutine ADDC5.

After the oversize items are configured into plane width strips, the list of strips is augmented by the bulk (palletized) items. Since only one pallet with no items adjacent can be placed on a strip, there is no need to call WIDFIT for bulk cargo. The addition of the bulk items to the list of strips is performed by ADDBLK.

Comments:

Forcing the algorithm to pack outsize cargo first insures that as much of the cargo bay of the C-5's as possible (for this algorithm) is actually packed with outsize material. It is possible to pack oversize and bulk material first on all plane classes. This would require the remaining cargo bay area of a C-5 to be consolidated into a single cargo bay. This option has not been explored.

Also, the cargo is packed in the order of its hazardous index. Currently, non-hazardous cargo has index 1, and hazardous, index 2. This can affect the solution when only empty planes are used. In this case all available aircraft may be utilized in lifting the non-hazardous cargo. If this poses a problem, the indices can be reversed in the data base.

4.2.2 OPTION

Purpose:

The purpose of the OPTION subroutine is to read in the user's solution strategy preferences, and a random number seed.

Arguments:

The one argument to OPTION is SEED, the random number seed.

Execution:

Currently, OPTION is set up to read the six input parameters from the file OPTIONS. The value of the cost function option, CSTOPT, determines which loading option will be used.

4.2.3 GETIT

Purpose:

This subroutine reads the cargo file and places the items in the proper cargo class. It also collects data on the total amount of square feet and weight present for each cargo class.
Arguments:

All array and variable names found in GETIT are described in Appendix 6, section 6.2.
Execution:

The execution is straight forward. First, all variables are initialized. After each line of the input file is read, the item a assigned to a cargo class depending on the values of HZD and OUT. These are the hazardous and size markers.
Comments:

For testing purposes the GETIT routine contains a call to a uniform random variable generator. The random variable indicates the record in the data base to be used for the next cargo item. While this is convenient for testing, it requires the use of a direct access file as the input file. Also, the integers in this input file are two bit integers, usually not supported by mathematical functions. To avoid this random record input, delete the ten lines beginning with the OPEN statements, ending with PRIOR=1.0, delete the three lines assigning values to NUM, HZD,and OUT, add an open statement for the new input file, un-comment the READ statement tagged #6, and add GO TO 6, prior to the statement labelled 10.

4.2.4 GETPL

Purpose:

This subroutine is responsible for reading the available aircraft file and determining the value of the ACL estimate used by WIDFIT. The estimate of the ACL for an aircraft class determines the maximum weight of a strip generated by WIDFIT for that aircraft class. The estimate is computed by the function LACL.
Arguments:

There are two arguments to GETPL. The first is a four element array for the ACL estimates for each of the aircraft classes, the last is a two element array for the widths of the wide and narrow bodied aircraft classes.
Execution:

After initialization, the aircraft file is read. Upon reading each aircraft

record, the data is stored in the array AIRCAP. After reading all the data, the function LACL is called to compute the ACL estimates.

4.2.5 LACL

Purpose:

This function determines an estimate of the ACL for an aircraft cargo class.

Arguments:

There are three arguments to LACL, a hazardous code, aircraft type code, and the number of aircraft in the class.

Execution:

The function currently contains statements to compute either the minimum ACL, the average ACL over an aircraft class, or a non-binding estimate. The decision as to which of these functions is desired is determined by the parameter, ACLOPT.

Comments:

The choice of the estimate of the ACL for each cargo class can affect the efficiency of WIDFIT. If the estimate is too low, the weight restriction prevents the placement of pieces on a strip, forcing more strips to be generated. However, a low estimate cannot prevent a piece from being placed on any strip. WIDFIT assumes that a single piece of cargo can always define a strip. A high estimate may create strips that cannot fit on some aircraft.

4.2.6 SORTER

Purpose:

This subroutine contains the calls to sort the wide and narrow item lists for each cargo class that are present in the cargo file.

Arguments:

All data for SORTER are in common, see Appendix 6, section 6.2.1

Execution:

The subroutine SORT is called for each of the wide and narrow item lists.

4.2.7 SORT

Purpose:

This subroutine performs a bubble sort on the input array. It is called by SORTER to provide WIDFIT with pointers, arranged in non-increasing item length, to the item lists.
Arguments:

The three arguments to SORT are the length of the input array, MAXITM; the array to be sorted, Y; and the pointer array, L.
Execution:

The pointers are initialized to the integers $1 \ldots MAXITM$, and a copy of the input array is made. This copy is used in sorting so the original array positions are not disturbed. The inner loop checks if two adjacent array elements are out of position, if so the continue code, $(M = 0)$ is set and the elements and pointers are swapped.

4.2.8 WIDFIT:

Purpose

WIDFIT performs a modified Next-Fit Decreasing Height bin packing. It's purpose is to coalesce individual pieces of cargo into a strip whose width is no more than the width of the cargo bay of the aircraft class to which the cargo may fit. An analogous task is trying to assign books to bookshelves so as to minimize the number of shelves required. This task is performed separately for each cargo class except the bulk classes. The assignment of cargo pieces to strips allows the next decision step, assigning strips to aircraft, to be formulated as an integer program.

While constructing strips, WIDFIT calls ADDPAX which determines an estimate of the number of passengers that can fit alongside the cargo on that strip. This information is stored in the common block for that strip class.

WIDFIT also determines the total length of all strips that it constructs. This is useful in measuring the efficiency of the strip generation. If the length of all the strips is not significantly lower than the length of all the items, then not many items can be placed adjacent to one another, or a different bin packing heuristic may be needed. It is also an indication of the number of narrow items in the cargo set; without many narrow items strips will consist of single items.
Arguments:

The arguments to WIDFIT are the hazardous index, the outsize index, the cargo bay width, and ACL, and the total length of strips generated.
Execution:

The items passed to WIDFIT are partitioned into a wide and a narrow item list. This segregation of the data was chosen because only one wide item may fit on a strip, while any number of narrow items can define a strip. Thus, each wide item is placed on a strip and then the narrow item list is checked for items that fit adjacent to the last item placed on the strip. If the narrow item is long enough so that more than one of the current wide item may fit alongside, without increasing the length of the strip, additional wide items are placed on the strip. This is the purpose of the first half of WIDFIT, (contained in the DO loop labelled 20).

Initially, the strip dimensions are that of the first item placed there. The loop which iterates over the narrow item list, loop 10, does so by means of two flags, MFLAG and JFLAG. These flags are updated to prevent iteration over the initial and terminal portion of the narrow item list, after those items have been assigned to strips. If a narrow item is found that fits on the current strip, its cursor in the cargo file is appended to the strip ID name, the number of those items is decremented, and if necessary the PACKED marker is set to true if it is the last of those items. Every unpacked narrow item is checked to see if it fits on the strip.

After this loop the members of the strip are fixed, and all that needs be done is to determine the number of pax that fit alongside the cargo and collect data totals.

The second half of WIDFIT, loop 50, performs the same task as loop 10, except that it tries to construct strips from the remaining narrow items.
Comments:

The choice of priority function, maximum, sum, or percentage of square foot utilization is made by the parameter, PRYOPT.

4.2.9 IMPORT

Purpose:

The IMPORT function determines a priority measure for the strip based on the priorities of the elements of the strip.
Arguments:

There are five dummy arguments for IMPORT. Depending on the option

code used, the variables have different interpretations.
Execution:

When priorities are assigned to items, the maximum priority of an item on a strip can define the strips priority. If all items have equal priority, or no priorities are assigned, a useful measure of strip priority is the ratio of square feet used by items on the strip to the area of the strip. Another measure is the sum of the priorities of items on the strip. The determination of the function used is made by the parameter, PRYOPT.

4.2.10 ADDPAX

Purpose:

This function determines the number of passengers that may fit alongside the pieces in a strip.
Arguments:

The three arguments to ADDPAX are the strip length and width, and the cargo bay width.
Execution:

If there is less than three feet of remaining width in the cargo bay, no people fit. If there is at least a three foot aisle but less than six feet, there is one passenger for each three feet of strip length. If there is more than six feet of aisle width, there are two passengers for each three feet of strip length, reflecting seating on both sides of the aircraft.

4.2.11 CNVRT

Purpose:

This subroutine converts the number passed to it to a four character representation of the same number.
Arguments:

The two arguments are the integer to be converted and the character string returned.
Execution:

The input integer is copied, each digit is converted to its EBCDIC code, and stored in a character of the output string.

4.2.12 LLOAD/GLOAD

Purpose:

LLOAD is a specially designed integer programming heuristic. It attempts to assign each strip to an aircraft by choosing a profit for the assignment of each strip to each aircraft. LLOAD is the shell which calls the profit and cost functions in an efficient manner.

Arguments:

The arguments to LLOAD are the number of aircarft in this class, NPLN, the hazardous index. IHZD, and the outsize index, IOUT.

The subroutine determines the maximum and second best profit attained by a strip on an aircraft, storing these values, in MXCOST and NXCOST, along with the strip and aircraft where they are achieved. Initially, this is performed by the nested loops 20, and 30. The profit of each strip is set to -1, an indication that the strip cannot fit on any aircraft.

The stopping rule, statement 40, is executed next. If all strips have profit -1, then the maximum profit is -1, so no strips fit on any aircraft. This is also a sign that more aircraft may be needed. If there are empty aircarft available, some of them are opened for loading. The number of aircraft pulled at any one time depends on the amount of cargo remaining and the cost option that is in affect. After augmenting the aircraft list execution returns to the beginning of the pricing stage.

Loop 40 decrements the aircraft capacities, and updates the passenger count. The best profit is then set to the second best profit. If the strip that achieves this new best profit is now infeasible on the plane where it should be placed, due to the placement of the last strip on the same plane, the best cost is set to -1. This allows loop 50 to find a new best and next best profit. Loop 50 only updates profits for all strips on the aircraft most recently updated, and all aircraft for those strips which became infeasible. It is not necessary to update the profits over all aircraft and strips after each strip is packed, except in special cases, discussed later.

The final block if in loop 50 updates the best profits. When a strip on a specific aircraft has a profit better than the current best and that strip does not define the current best, all the variables involved with strip placement are updated. If that strip does define the current best profit, only the profit and placement variables are updated. This prevents one strip being considered the best and next best strip simultaneously and then loaded

twice. Finally, the second best profit and placement variables are updated.
Comments:

LLOAD is a shell to correctly call the profit and cost functions. The other version, in GLOAD (Global Loader), performs the profit updates over all aircraft and strips after each placement. While this extra execution is not needed for some of the profit and cost functions, other functions require more global information. These profit and cost functions will be described in the next section.

4.2.13 PROFIT

Purpose:

The PROFIT function determines a measure of a good location for a strip. It is thus dependent on strip dimensions, possibly all unpacked strip dimensions, individual aircraft dimensions, and possibly the remaining dimensions of the aircraft class. This is in contrast to the IMPORT function which is only dependent on strip parameters. The higher the value of the profit function, the better that particular assignment.
Arguments:

The arguments to PROFIT are the strip parameters, the unpacked strip totals, the aircraft dimensions, and the cargo space remaining in the aircraft class.
Execution:

There are two functions imbedded within PROFIT. The choice of function is determined by PRFOPT. In the first function the value of the profit for placing strip A on plane B is the minimum of the ratios, A's length to B's remaining cargo bay length, and A's weight to B's remaining ACL, times the priority of A. The second function determines which resource, length or ACL, of the aircraft is scarcest and assigns a profit corresponding to the percentage utilization of the strip's other resource times its priority.
Comments:

The choice of profit function is crucial to the efficient solution of the aircraft loading problem. The function has been designed to allow the incorporation of a diverse set alternatives. This is the reason for the unused arguments passed to PROFIT, it allows flexibility in development of better functions.

4.2.14 PENLTY:

Purpose:

The PENLTY function penalizes the use of limited aircraft resources. The greater the resource consumption by a strip the larger its PENLTY value should be. This value depends on the strip parameters, possibly the unpacked strip parameters, the individual aircraft capacity, and possibly the remaining aircraft class' capacity.

Arguments:

The arguments are the strip parameters, the unpacked strip parameters, the individual aircraft capacity, and the remaining aircraft class' capacity.

Execution:

There are many possible penalty functions to choose from, each with different solution qualities. One choice is to penalize an assignment by the maximum of the two aircraft resources that are consumed, i.e. the penalty for placing strip A on plane B is the maximum of the ratios, A's length to B's remaining cargo bay length, and A's weight to B's remaining ACL. Another possible penalty function is to penalize the consumption of the aircraft class' resources instead of an individual aircraft's. Three penalty functions are coded, the choice of which to use is made via CSTOPT.

Comments:

Both functions mentioned above are coded and chosen by an option code. The arguments to PENLTY were chosen to be broad enough to allow easy implementation of any choice of function type.

4.2.15 ADDBLK

Purpose:

This subroutine appends the list of bulk cargo to the list of strips generated by WIDFIT. This avoids processing the bulk cargo through WIDFIT, since it is known that a pallet requires the width of a narrow-bodied aircraft.

Arguments:

The only argument to ADDBLK is the hazardous cargo marker.

Execution:

The nested loops create a strip with the characteristics of the pallet in question at the end of the list of strips.

4.2.16 ADDC5

Purpose:

ADDC5 constructs two dummy cargo bays from each C-5 cargo bay, after the C-5's have been loaded with outsize cargo. Only C-5's which have been used are split into dummy cargo bays.

Arguments:

The arguments to ADDC5 are the hazardous cargo marker, the new number of narrow-bodied aircraft, and width and ACL estimates for the narrow bodied aircraft class.

Execution:

For each C-5 aircraft, the subroutine creates two cargo bays with one-half the ACL, one-half the width, and the same length. The width of the cargo bay used for WIDFIT is the smaller of the two widths. The function LACL is called again to update the estimate of the ACL for the aircraft class.

Comments:

If it is allowed that an empty C-5 be loaded with only oversize and bulk cargo, changes must be made in two subroutines. In ADDC5, the variable NUSED(IHZD,1) should be replaced with NUMPL(IHZD,1), and the UTIL array should be set to false for those C-5's not yet utilized. The other changes occur in the report writer. Here in the loop labelled 17 and the statement preceeding it, the array NUSED is replaced with NUMPL. An important distinction for these changes is the USED array indicates the number of aircraft made available by GETPL and ADDAIR, it does not indicate the number of aircraft that have had strips assigned them.

4.2.17 REPORT

Purpose:

REPORT writes an output file of relevant to the solution of the aircraft loading problem and its quality.

Arguments:

All data used by REPORT are in common blocks.

Execution:

The subroutine first prints the total amount of square feet and ACL available for each aircraft class, then the equivalent number of empty air-

craft that the total corresponds to. Next, the dummy C-5 cargo bays are consolidated, and the remaining individual aircraft capacities are printed in the order they appear in, in the input file.

As mentioned previously, if C-5's are allowed to carry only oversize and bulk cargo changes are necessary in this subroutine. This changes, discussed in section 4.2.16, result from the way aircraft data are stored in memory. For the narrow-body aircraft, the aircraft present initially are stored first, then the dummy C-5 aircraft are added and finally any empty aircraft are placed at the end of the list. The placement of the dummy C-5 cargo bays in the middle of the list leads to some gymnastics to report aircraft capacities. Essentially, the number of C-5's added, and the original number of narrow bodied aircraft must be known. The number of C-5 aircraft added is found in either NUMPL or NUSED depending on the restrictions on how C-5's are loaded with non-outsize cargo. Currently, a C-5 which has been loaded with some outsize cargo may be filled with oversize and bulk cargo. If a C-5 is available for outsize loading and has no outsize cargo packed on it, it may not be used for shipping oversize or bulk cargo.

The total amount of cargo, and load equivalents, for each class is reported next. Following these totals, the unpacked strips are reported, along with the total amount of unpacked cargo. The subroutine DECIFR is used to determine the pieces of cargo that are in each unpacked strip.

Included in the putput file are some lower bounds on the solution values. The linear programming value for a single aircraft class is equal to the maximum of the equivalent plane loads when only one constraint is binding, see PDRC 88-03. Thus the LP solution for the number of C-5's required to lift the outsize cargo is easily computed. After subtracting away the amount of over sized and bulk cargo placed on C-5's, the remaining cargo must all be moved by C-141's, so the LP problem is again trivial. This solution is also reported. Note, that the length bound is computed by the length of the strips, not individual items.

Also included is a measure of inherent waste in the bin packing. If no item can fit alongside a wide item, then the square feet alongside that wide item must be wasted in any packing. The total amount of all square feet found in this way is reported as a percentage of a cargo bay. This bound suffers immensely because one narrow item can feasibly be placed alongside any wide item. Thus one narrow item can count against hundreds of wide items.

4.2.18 DECIFR

Purpose:

This subroutine takes a strip ID character string and returns an array containing the cursors to the item lists.

Arguments:

There are two arguments to DECIFR, the character string, ID, and the array of cursors, ITEM. There is an inherent limit of ten items in the dimension of ITEM.

Execution:

Every four consecutive elements of ID, beginning at the second, define a cursor position. These characters are passed to NUM, which converts the characters back to integers, stored in an element of ITEM.

4.2.19 NUM

Purpose:

This subroutine converts the four digit character string, A, into an integer representation, K.

Arguments:

The two arguments to NUM are the four element character string, A, and the integer, K.

Execution:

The subroutine uses the intrinsic FORTRAN function ICHAR to convert each character to its integer representation.

4.2.20 ADDAIR

Purpose:

This subroutine adds additional empty aircraft to the available aircraft list when all current aircraft are full, and the aircraft option code allows it. Note, that this subroutine was not in earlier versions of ALPE.

Arguments:

There are eight arguments to ADDAIR. They are: the aircraft type, the cost option, the hazard code, the pointers to the beginning and end of the aircraft list, the total weight and length of cargo remaining to be packed, and the total remaining amount of length and ACL in the aircraft fleet.

Execution:

The subroutine checks the aircraft type and cost option before determining the number of aircraft of which type to add. The number of aircraft to add may be the LP bound for the remaining cargo, or a fixed number depending on the parameters NUMFIX and IADD, and the cost option.

Once the number of aircraft to add has been determined, the proper totals are updated as are the pointers to the aircraft list.

Comments:

This subroutine was added to allow for PC compilation of ALPE; previous versions illegally jumped in and out of IF blocks. It also allows for more flexible allocation of aircraft assets.

5 COMPILING AND RUNNING ALPE

5.1 Files

All the FORTRAN code for ALPE is in the file ALPE.FOR. The input files required to run ALPE are the cargo data file, an options file, and possibly a seed and an aircraft file. The seed file is necessary for using the random number generator in GETIT, if GETIT is modified to read an ordinary input file, the seed file is no longer needed. The aircraft file may be necessary depending on the choice of PLNOPT.

5.1.1 Option File

The option file requires the specification of six option codes. The first code specified is the priority code. This determines if the priority assigned to a strip will be the maximum of the items' priorities, their sum, or the square feet utilization rate. The second code is the cost option. The three choices here indicate no cost, a cost computed over individual aircraft, and a cost determined by the entire class of aircraft. Choosing the last option requires GLOAD, while the other two use LLOAD. The profit option determines which of two profit functions will be used. The first function is described in section 3.4. The other function checks which resource is scarcest and computes the profit as the percentage utilization of the other resource. Hence, this function encourages use of the resource which is most plentiful.

The next option code is the ACL option. This code determines how the ACL estimate is computed in LACL. The options are to choose the

minimum ACL, the average ACL, or a non-binding estimate. Empirically, the minimum estimate is undesirable. The fifth option code is an output option. This option regulates the amount of information printed to the output files. At the highest level, the output file contains the remaining capacity of each aircraft, and each strip that was not packed, in addition to the totals provided by the summary output. Also, the name, length, weight, and priority of strips generated can be printed to a file for additional analysis.

The last option code is the plane option which has three possible values. Choosing the first option allows the program to read the aircraft capacities from a file. In this way partially filled aircraft may be input to the algorithm. The other two options assume that all aircraft are empty. The first of these allows a reserve of additional empty aircraft to ensure the loading of all items. The second allows only the LP bound number of aircraft for each cargo class.

5.2 Compilation

The ALPE code has been developed under the IBM VM/IS operating system and compiled using the VS FORTRAN version 2 compiler. The nature of the model requires the storage of vast quantities of information.

In one test run, almost 1700 strips were generated. The storage of the initial cargo data and pointers alone requires 29,000 bytes of memory. The entire memory requirement for this largest case exceeded 1 million bytes. If the number of aircraft in any class is limited to 25, the number of strips to 200, and the number of records in any class to 50, the memory allocation necessary is just under 133,000 bytes.

The program itself requires just over 114,000 bytes of memory. Most of this space is used by LLOAD, GLOAD, WIDFIT and REPORT, which consume almost two-thirds of the space.

Most of this memory allocation is wasted space. The fixed dimensionality of common blocks in FORTRAN requires that each cargo class consume the same amount of memory. This assumption is unfounded in this application, where there often are void cargo classes, i.e. if no hazardous cargo is present the amount of memory for cargo can be reduced by at least one-half.

The decision to redesign the data structures was predicated on the discussion in December with ORNL, where the option of changing the code to Ada was discussed. The current data structures should allow easier transition to a linked list structure.

Also, PC compilers, such as MICROSOFT's FORTRAN compiler, allow for compression of integers from four to two byte representations, and have adjustable size memory addresses. That is the word size of memory addresses can be set at 32 bytes per address in the huge memory model. These options may allow this code to run on a PC.

APPENDICES

6 APPENDIX

1. DICTIONARY for ALPE COMPUTER CODE

6.1 PARAMETERS

MAXINS	maximum number of items in any cargo class
MAXNUM	maximum number of aircraft that share width, and hazardous codes
MAXSTR	maximum number of strips in any cargo class
MAXTYP[4]	the number of different cargo bay widths
NUMFIX	determines if long or short aircraft lists are used
IADD	determines the incremental length of the aircraft lists
WIDOUT	one-half the width of the cargo bay of the aircraft that transports the outsize cargo
WIDOVR	one-half the width of the cargo bay of the narrow body aircraft type

6.2 COMMON BLOCKS

6.2.1 Item and Strip Data

The cargo common block, ITEM, consists of eight arrays. The CARGO array contains five indices; a hazardous index, outsize index, wide/narrow index, record index, and a dimension index. The four dimension indices correspond to length, width, weight, and priority.

The NCARGO array counts the number of records in each of the cargo classes, subdivided by width. The NUMCAR array stores the number of individual pieces of cargo in each record. The CARPTR array stores pointers to the cargo array, sorted into non- increasing length.

The STRIPS array contains six fields for each strip. These fields are the length, width, weight, priority, passengers and square feet of the strip. The NSTRIP array counts the number of strips in each cargo class. The STRPID array contains the identifiers for all items on that strip. The

[4]This parameter cannot be changed without major modification of the code.

STRPKD array is a boolean array indicating whether each strip has beeen packed.

6.2.2 Other Common Blocks

Aircraft (AIRCRF)

AIRCAP	a four dimensional array indexed by aircraft type, hazardous code, individual plane ID, dimension (1=width,2=ACL,3=length,4=pax)
AIRSQ	available square feet for each of the four aircraft classes
AIRWT	available ACL for each of the four aircraft classes
NUMPL	the number of planes in each of the four aircraft classes
EMPTY	the number of empty aircraft
USED	the number of aircraft of each class utilized
UTIL	a boolean variable indicating whether each aircraft has been utilized

Totals (TOTALS)

NUMTOT	the number of strips in each of the six cargo classes
TOTLN	the length of all the strips in each cargo class
TOTSQ	the square feet used by all the strips in each cargo class
TOTWT	the weight of all the items in each cargo class
OVONC5	the weight and length of oversize and bulk cargo on C-5 aircraft

Options (OPT)

PRYOPT	the priority function option, used in IMPORT
CSTOPT	the cost function option, used in PENLTY
LODOPT	the loading subroutine option, used in MAIN
PRFOPT	the profit function option, used in PROFIT
ACLOPT	the ACL estimate option, used in LACL
OUTOPT	the output level option, used in REPORT, WIDFIT, and loaders
PLNOPT	the input option for aircraft, used in GETPL

Waste (WAIST)

NARWID	the narrowest width item for that cargo class
WASTE	the total amount of square feet not usable in any packing
SHORT	the shortest item, used to determine if a plane is full
LIGHT	the lightest item, used to determine if a plane is full

6.3 MAIN Program

ACL	the ACL estimate for each aircraft class
AIRWID	the width of the cargo bay of the aircraft
AIRWD2	one-half the cargo bay width of the wide-bodied aircraft

6.4 GETIT (Get Items)

All variables are defined in the common block section, or are self-explanatory.

6.5 GETPL (Get Planes)

NN	the number of aircraft to make available if PLNOPT is 2
NUMEM1/2	the number of empty aircraft of each type
X1/2	the equivalent plane load estimates for square feet and weight

6.6 LACL

All variable name are self-explanatory.

6.7 SORTER

All variables are defined in the common block section

6.8 SORT

This subroutine is a bubble sort procedure. The array to be sorted is Y. Pointers to Y are stored in L. The length of Y is passed as MAXITM. The array Y is copied to X so that the original placment of the Y data is not disturbed.

6.9 WIDFIT (Width Fit)

A	the character representation of the item index
FIRCHR	the index of STRPID where the current item name begins
IFLAG	the index of the last unpacked item
JFLAG	the index of the last unpacked item
KFLAG	the index of the first unpacked item
KK	determines if current index is the first execution of this loop
MFLAG	the index of the first unpacked item
NACROS	the number of narrow items that fit across the cargo bay
NPTR	pointer to the narrow items
NSTRP()	the strip currently being packed
NPUT	the actual number of items to be placed on the strip
NUMPKD	the number of narrow items packed
NUMPUT	the number of items that fit according to NVERT and NACROS
NUMSTR	the number of strips generated
NVERT	the number of items that fit "vertically" in the strip
NWAIT	the number of items that fit according to the weight restriction
NXTCHR	the index of STRPID of the last character of the item name
PLNWID	the maximum width of a strip
PLNACL	the maximum weight of a strip
TOTLEN	the length of all the strips generated in this pass
WIDPKD	the number of items remaining to be packed in the current record
WPTR	the pointer to the current record

6.10 ADDPAX (Add passengers)

LENGTH	the length of the strip
PWIDTH	the cargo bay width
WIDTH	the strip width

6.11 LLOAD/GLOAD

ACL	the index of the ACL in the AIRCAP array (=2)
COST	the cost of the strip on the current aircraft
COST1()	the current best cost of that strip
ISTAR	the item that achieves MXCOST
JSTAR	the aircraft that achieves MXCOST
LEN (length)	the index of the length in the AIRCAP array (=3)
LOW	the index of the first unfilled plane
MITEM	the strip that achieves MXCOST
MWHERE	the aircraft where MXCOST is achieved
MXCOST	the best cost
NITEM	the strip that achieves NXCOST
NPLN	the number of aircraft
NUMSTR	the number of strips to be packed
NXCOST	the second best cost
NWHERE	the aircraft where NXCOST is achieved
PRIOR()	the priority of that strip
STOTLN	the total length of remaining cargo
STOTWT	the total weight of remaining cargo
WHERE()	the aircraft ID where the best cost for that item is achieved

6.12 PROFIT

The arguments to PROFIT are dummies; however, the names are mnemonic for the simplest profit function used. In this case the prefix 'P' denotes plane capacities, 'I' denotes item sizes, and IMPORT the items priority.

6.13 PENLTY (Penalty)

The arguments are dummies, and reflect plane and item dimensions.

6.14 CONVRT (Convert)

These names are arbitrary.

6.15 ADDC5

The arrays found here are defined in the common block section of this appendix.

6.16 REPORT

The arrays, and variable names in REPORT are the names found in the common block section. Those arrays beginning with 'U' denote the total amount of each dimension of each cargo class that is unpacked. The NAMES array stores the indices of the items in that strip.

6.17 DECIFR (Decifer)

These names are self-explanatory.

6.18 NUM (Number)

These names are arbitrary.

6.19 OPTION

All the variable names in OPTION are explained in the common block OPT.

6.20 IMPORT

The variables are dummies, but are mnemonic for the third priority option. The variable WIDTH refers to the cargo bay width, to determine square foot utilization.

6.21 ADDBLK

All arrays are in the ITEM or TOTALS common blocks.

6.22 ADDAIR

A	a dummy for the total amount of cargo bay length opened
B	a dummy for the total amount of cargo bay ACL opened
EQVLEN	the number of equivalent plane loads of remaining cargo by length
EQVWT	the number of equivalent plane loads of remaining cargo by weight
NADD	the number of empty planes to add to the available list
XTOTLN	the total length of remaining cargo
XTOTWT	the total weight of remaining cargo

7 APPENDIX

1. TABLES of COMPUTATIONAL RESULTS

DATA SET ONE

14	outsize strips	LP bounds:	C-5 = 5.26
947	oversize and bulk strips		C-141 = 103.13

Table 1: Number of Aircraft by Option Code (C5, C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	6,108	6,106	6,**109**	6,108	6,107	6,108
	2	6,106	6,106	6,106	7,104	6,106	6,107
	3	6,108	6,106	6,107	7,*103*	6,106	6,106

Table 2: Run Time by Option Code (in seconds)⋆

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	71.1	69.5	70.3	72.0	70.2	70.5
	2	74.0	71.1	72.0	78.7	73.1	77.3
	3	130.0	131.6	130.2	144.9	135.0	140.1

⋆These run times used short available aircraft lists, i.e. NUMFIX=1, IADD=5.

DATA SET ONE

Number of aircraft available: C-5 = 5 C-141 = 99

Minimum upper bound: C-5 = 6 C-141 = 103

Table 3: Ratio of Minimum Percent of Cargo Lifted to Expected Percentage (C5,C141)

		(PRIORITY Code, PROFIT Code)					
		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	1,.953	1,.974	1,.942	1,.953	1,.964	1,.953
	2	1,.974	1,.974	1,.985	.888,.985	1,.974	1,.974
	3	1,.953	1,.974	1,.974	.888,.996	1,.985	1,.985

DATA SET TWO

54	outsize strips	LP bounds:	C-5 = 23.71
496	oversize and bulk strips		C-141 = 58.30

Table 4: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	42,0	43,0	43,0	42,0	42,0	42,0
	2	43,0	43,0	43,0	42,0	43,0	43,0
	3	43,0	43,0	43,0	42,0	43,0	43,0

Table 5: Run Times by Option Code (in seconds)†

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	67.3	74.6	83.0	70.9	67.0	80.0
	2	69.6	59.6	70.4	58.8	48.6	89.5
	3	487.2	477.0	509.7	506.8	529.9	578.3

†These runs used short aircraft lists.

DATA SET TWO

Number of available aircraft: C-5 = 24 C-141 = 57

Minimum upper bound: C-5 = 42 C-141 = 0

Table 6: Ratio of Minimum Percent of Cargo Lifted to Expected Percentage (C5 only)‡

		(PRIORITY Code, PROFIT Code)					
		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	*	1	1	*	*	*
	2	1	1	1	*	1	1
	3	1	1	1	*	1	1

‡An asterisk denotes more than the expected amount was lifted.

Table 7: Number of C-141 aircraft used by Option Code

Due to the large amount of unused space on C-5 aircraft, this is the only data set where all bulk and oversize cargo was packed on less C-141 aircraft than were available.

		(PRIORITY Code, PROFIT Code)					
		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	29	27	27	29	28	29
	2	25	26	25	26	24	24
	3	27	27	26	26	26	26

DATA SET THREE

2	outsize strips	LP bounds:	C-5 = 0.34
655	oversize and bulk strips		C-141 = 83.82

Table 8: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	1,86	1,86	1,86	1,86	1,86	1,86
	2	1,86	1,86	1,86	1,86	1,86	1,86
	3	1,86	1,86	1,*85*	1,86	1,86	1,86

Table 9: Run Times by Option Code (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	31.7	31.7	31.8	34.4	32.6	32.5
	2	139.2	97.0	107.4	83.3	93.6	164.6
	3	689.5	661.5	687.5	688.3	642.4	687.7

DATA SET THREE

Number of aircraft available: C-5 = 1 C-141 = 75

Minimum upper bound: C-5 = 1 C-141 = 85

Table 10: Ratio of Minimum Percent of Cargo Lifted to Expected Percentage (C5,C141)

		(PRIORITY Code, PROFIT Code)					
		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	1,.929	1,.941	1,.941	1,.941	1,.941	1,.941
	2	1,.941	1,.941	1,.963	1,.941	1,.952	1,.952
	3	1,.952	1,.941	1,.963	1,.941	1,.941	1,.952

DATA SET FOUR

3	outsize strips	LP bounds:	C-5 = 0.66
1674	oversize and bulk strips		C-141 = 174.63

Table 11: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	1,*180*	1,*180*	1,181	1,*180*	1,181	1,181
	2	1,181	1,181	1,181	1,181	1,181	1,**182**
	3	1,181	1,181	1,*180*	1,181	1,181	1,181

Table 12: Run Times by Option Code (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	163.8	166.1	165.9	164.4	164.1	165.7
	2	1198.2	1295.0	1348.7	1350.6	1257.4	1382.7
	3	11054.	10996.	11118.	11439.	11656.	11494.

DATA SET FIVE

16 outsize strips LP bounds: C-5 = 4.79

550 oversize and bulk strips C-141 = 68.6

Table 13: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	7,66	7,66	7,66	7,66	7,66	7,66
	2	7,66	7,66	7,66	7,66	7,66	7,66
	3	7,66	7,66	7,66	7,66	7,66	7,66

Table 14: Run Time by Option Codes (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	30.8	30.3	31.2	31.3	30.9	31.4
	2	85.9	84.9	91.5	95.0	108.0	103.8
	3	432.6	417.9	448.9	464.5	432.3	470.5

DATA SET FIVE

Number of aircraft available: C-5 = 4 C-141 = 61, or 58◇

Minimum upper bound: C-5 = 7 C-141 = 66

Table 15: Ratio of Minimum Percent of Cargo Lifted to Expected Percentage (C5,C141)

		(PRIORITY Code, PROFIT Code)					
		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	1,.855	1,.855	1,.855	1,.855	1,.866	1,.855
	2	1,.797	1,.785	1,.797	1,.808	1,.797	1,.797
	3	1,.808	1,.797	1,.797	1,.808	1,.797	1,.797

◇When two numbers are present the first is the number of aircraft for COST = 1, the second for the rest. This occurs when hazardous cargo is present.

DATA SET SIX

14	outsize strips	LP bounds:	C-5 = 3.95
1071	oversize and bulk strips		C-141 = 111.45

Table 16: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	5,114	5,114	5,114	5,114	5,114	5,114
	2	5,114	5,114	5,114	***4,116***	***4,116***	5,*113*
	3	5,*113*	5,114	5,114	5,114	5,114	5,*113*

Table 17: Run Times by Option Code (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	74.9	74.1	74.8	77.5	76.6	76.7
	2	408.9	395.1	458.0	438.4	390.9	399.0
	3	2924.	3004.	3056.	3170.	3209.	3251.

DATA SET SIX

Number of aircraft available: C-5 = 4 C-141 = 113, 105

Minimum upper bound: C-5 = 4 C-141 = 113

Table 18: Ratio of Minimum Percent of Cargo Lifted to Expected Percentage (C5,C141)

		(PRIORITY Code, PROFIT Code)					
		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	.822,*	.822,*	.822,*	.911,.989	.911,*	.911,*
	2	.838,.969	.822,.969	.838,.969	1,.958	1,.947	.855,.958
	3	.838,.969	.911.960	.838,.965	.859,.965	.855,.956	.949,.956

DATA SET SEVEN

55	outsize strips	LP bounds:	C-5 = 12.86
1241	oversize and bulk strips		C-141 = 139.81

Table 19: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	15,134	15,*133*	15,134	15,134	15,**135**	15,134
	2	15,*133*	15,134	15,134	15,*133*	15,134	15,134
	3	15,134	15,134	15,134	*14*,**138**	15,**135**	15,134

Table 20: Run Times by Option Code (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	112.1	139.4	124.0	127.1	117.8	133.3
	2	877.3	758.3	590.3	739.9	743.6	1019.
	3	4980.	5524.	5193.	5257.	5157.	5308.

DATA SET SEVEN

Number of aircraft available: C-5 = 13 C-141 = 127

Minimum upper bound: C-5 = 14 C-141 = 133

Table 21: Ratio of Minimum Percent of Cargo Lifted to Expected Percentage (C5,C141)

		(PRIORITY Code, PROFIT Code)					
		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	.965,.953	.983,.952	.983,.965	.983,.948	.983,.946	.983,960
	2	.949,.958	.983,.959	.983,.969	.965,.971	.983,.950	.983,.950
	3	.949.954	.983,.956	.983,.970	*,.932	.983,.954	.945,.974

DATA SET EIGHT

2	outsize strips	LP bounds:	C-5 = 0.33
1303	oversize and bulk strips		C-141 = 137.74

Table 22: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	1,142	1,142	1,142	1,142	1,142	1,142
	2	1,142	1,142	1,142	1,142	1,142	1,144
	3	1,142	1,142	1,142	1,142	1,142	1,142

Table 23: Run Times by Option Code (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	103.4	103.2	103.4	105.0	105.0	105.0
	2	633.6	621.7	571.8	716.6	925.5	838.3
	3	5028.	5059.	5240.	5123.	5136.	5152.

DATA SET EIGHT

Number of aircraft available: C-5 = 1 C-141 = 130, 129

Minimum upper bound: C-5 = 1 C-141 = 142

Table 24: Ratio of Minimum Percent of Cargo Lifted to Expected Percentage (C5,C141)

		(PRIORITY Code, PROFIT Code)					
		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	1,.978	1,.975	1,.981	1,.979	1,.979	1,.979
	2	1,.975	1,.975	1,.976	1,.975	1,.974	1,.957
	3	1,.975	1,.976	1,.977	1,.976	1,.973	1,.976

DATA SET NINE

15 outsize strips LP bounds: C-5 = 4.68

430 oversize strips (no bulk) C-141 = 86.86

Table 25: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	7,87	7,88	7,88	7,86	7,87	7,88
	2	7,86	7,87	7,*85*	*6*,90	*6*,91	7,**94**
	3	7,87	7,86	7,87	*6*,90	*6*,92	7,87

Table 26: Run Times by Option Code (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	27.6	23.6	23.3	24.2	24.5	24.6
	2	128.0	130.9	128.4	126.8	128.5	147.6
	3	464.8	493.2	494.2	483.6	498.2	556.0

DATA SET TEN

4	outsize strips	LP bounds:	C-5 = 1.1
260	oversize strips (no bulk)		C-141 = 47.64

Table 27: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	2,47	2,47	2,47	2,47	2,47	2,47
	2	2,47	2,47	2,47	2,47	2,47	2,47
	3	2,47	2,47	2,47	2,47	2,47	2,47

Table 28: Run Times by Option Code (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	15.5	15.9	15.5	13.7	13.7	13.6
	2	31.6	30.1	30.9	26.0	28.2	29.1
	3	92.6	70.3	74.6	75.7	77.5	72.6

DATA SET ELEVEN

39	outsize strips	LP bounds:	C-5 = 12.64
510	oversize strips (no bulk)		C-141 = 81.60

Table 29: Number of Aircraft by Option Code (C5, C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	*16* ,78	18,72	17,74	*16* ,78	18,72	18,72
	2	18,72	18,**73**	18,*71*	18,72	18,72	17,75
	3	18,72	18,72	18,**73**	17,75	18,72	18,**73**

Table 30: Run Times by Option Code (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	15.5	15.9	15.5	13.7	13.7	13.6
	2	31.6	30.1	30.9	26.0	28.2	29.1
	3	92.6	70.3	74.6	75.7	77.5	72.6

DATA SET TWELVE

15 outsize strips LP bounds: C-5 = 3.98

366 oversize strips (no bulk) C-141 = 58.50

Table 31: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	5,58	5,58	5,58	5,58	5,58	5,57
	2	5,58	5,58	5,58	5,58	5,58	5,58
	3	5,58	5,58	5,57	5,57	5,58	5,57

Table 32: Run Times by Option Code (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	18.1	18.1	18.1	18.6	18.4	18.5
	2	41.3	38.5	39.2	44.9	38.5	43.6
	3	137.1	139.8	139.8	147.0	141.3	147.4

DATA SET THIRTEEN

26	outsize strips	LP bounds:	C-5 = 5.58
204	oversize strips (no bulk)		C-141 = 33.78

Table 33: Number of Aircraft by Option Code (C5,C141)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	6,35	6,*34*	6,*34*	6,*34*	6,*34*	6,*34*
	2	6,35	6,35	6,35	7,32	7,32	6,35
	3	6,35	6,35	6,35	7,32	7,32	6,35

Table 34: Run Times by Option Code (in seconds)

(PRIORITY Code, PROFIT Code)

		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	11.6	11.6	11.7	11.6	11.7	11.6
	2	17.4	16.7	18.1	18.1	18.2	18.7
	3	44.0	43.4	42.7	44.6	44.4	43.7

DATA SET EIGHT

Comparison of run times with long and short aircraft lists, aircraft numbers were unchanged.

Table 35: Run Times by Option Code (in seconds, short aircraft lists)

		(PRIORITY Code, PROFIT Code)					
		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	118.4	117.4	117.6	118.6	118.7	118.5
	2	118.7	140.8	126.5	123.6	146.3	153.2
	3	230.2	248.5	235.1	279.5	503.7	357.2

Table 36: Run Times by Option Code (in seconds, LP length aircraft lists)

		(PRIORITY Code, PROFIT Code)					
		(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)
COST	1	103.4	103.2	103.4	105.0	105.0	105.0
	2	633.6	621.7	571.8	716.6	925.5	838.3
	3	5028.	5059.	5240.	5123.	5136.	5152.

PROGRESS REPORT

Georgia Tech Aircraft Loading Project

April -June 1988

Activities Performed:

Continuation of accomplishments of PDRC 88-03

Since the delivery of PDRC 88-03, the research effort at Georgia Tech centered on extending the results of that report. Specifically, Theorem 4 of PDRC 88-03, which provided an easy solution methodology for two planes, and the Next-Fit heuristic were attacked.

Extension of Loading Heuristics

One of the hypotheses of Theorem 4 was that the binding constraints for both plane types are known. However, when the binding constraints are not known, the optimal solution is still easy to find. This is true because the assumption that the binding constraints are known only insures primal feasibility. The dual weights that are constructed are always dual feasible. Thus, to solve the LP problem all that needs to be done is solve the sorting problem for each pair of constraints, and take the maximum solution value as the answer. The maximum is an optimum because the solutions are dual feasible, and Theorem 4 guarantees one such solution must be optimal.

Another extension considered was that of incorporating more than two plane types in a solution methodology. Here there are three possibilities of solution strategies. The selection of proper dual weights in the proof of Theorem 4 relied on a "bumping" argument, i.e. if constraint i for plane type 1 was changed by a small amount, how much, of which item would be bumped to plane type 2. An analogous argument must be made in the multiple plane type case; the major difference being that each bumped item causes a ripple effect through each smaller plane type.

One method of choosing dual weights would be to use the results of Theorem 4 with multiple planes. This would be equivalent to the use of a rolling-horizon type model, where the current method would be applied to plane types i and i+1, the items assigned to plane type i removed and the method reapplied with i= i+1 until no more items are left. This has the advantages of being easy to code and very fast.

Another alternative would be to use the two plane type method with the plane type currently being packed and the smallest plane type to be used. The justification for this method is that as a fractional item is bumped from the current plane type, it bumps an item from the next plane type, on down the line until an item is bumped to the smallest plane type. Thus, the net effect will only be felt in the smallest plane type, but the size of the effect depends on all the items that are bumped.

With both of these methods knowledge of the binding constraints for each plane type would increase the accuracy of the heuristic. As the number of plane types grows it becomes computationally inefficient to try all pairs of constraints. However, since both of these procedures are not optimal, it seems that a reasonable guess at the binding constraints could be used. Currently, the max-sum model of PDRC 88-03 is used to determine which constraints will be considered binding, i.e. for plane type j, $\text{argmax}_k\{ \Sigma_i S_{ki}/C_{kj} \}$ is considered binding, where the sum ranges over those items not yet packed. Code for both of these methodologies has been written and transferred to the IBM 9375.

Any optimal sorting scheme will probably have to take into account all the interactions of bumping, making analysis much more difficult. Also, any heuristic that used a guessing routine to determine the binding constraints cannot guarantee primal feasibility.

Attention was also given to the Next-Fit heuristic. Since sorting by size ratios worked so well for the LP model, it was incorporated into the Next-Fit routine. As with the extensions of Theorem 4 to multiple plane types, here it seemed appropriate to use the max-sum model to estimate the binding constraints and sort by ratios prior to entering the Next-Fit heuristic.

Also, the algorithms of PDRC 88-03 were transferred to the IBM 9375 and the groundwork for the development of an algorithm that would address the problems of Level 4 data was begun.

Incorporating Level 4 Data

The incorporation of Level 4 data in the TUCHA and SRF is currently being explored and work on an algorithm to pull that data out of the raw data databases is underway and will not be difficult.

However, in order to meet our objectives, one of which is to identify hazardous cargo, is difficult given the limited information contained in the TUCHA and SRF (length, width, height, square feet, weight, and cube, and a Cargo Category Code which only indicates if cargo is hazardous but not where it could be placed). Also, even if a list was provided listing incompatible combinations

of cargo and limitations on their placement, it is not known how to recognize such items in the TUCHA or SRF where equipment is only identified through a highly abbreviated description field which allows for multiple names for the same item (e.g. "TRK CGO 5.0 TON" and "5 TON CARGO TRUCK").

One last concern about Level 4 data is about the sheer amount of data to be processed. Efforts are being made to make the data processing algorithm run more smoothly but until this is accomplished, an algorithm to preprocess Level 4 data will take even longer than the 14 hours mentioned in PDRC 88-03 for Level 3 data.

Inorder to help us realize the additional complexities of packing with Level 4 data, the procurement of the following references for our use would be helpful:

AFR 71-8/ TM 38-236 Preparation of Hazardous Material for Military Air Shipment

MACR 55-1 vol 1	Airlift Operations
MACR 55-1 vol 3	Tactical Airlift Operations
MACR 55-2	C-5 Airlift Operations
MACR 55-4	C-141A Configuration/ Mission Planning
MACR 55-26	C-5A Configuration/ Mission Planning
MACR 55-47	C-130 Configuration/ Mission Planning.

Applications of Artificial Intelligence to Plane Loading

Introduction to Artificial Intelligence Applications

It is hoped that applying Artificial Intelligence (AI) techniques to the problem of loading aircraft will result in (1) more efficient ways to handle the list of items to load, (2) modelling of human experts who defy common computer loading techniques in obtaining clever plane loads, and (3) adaptability to changing conditions. This section will discuss ideas examined in the early stages of the employment of AI and the results to date which include a working prototype of an algorithm which will recognize predefined patterns in a cargo list in order to aid in load planning.

What is Artificial Intelligence

Artificial Intelligence is defined by Elaine Rich as "The study of how to make computers do things at which, at the moment, people are better.[1] Computers have traditionally performed in areas where they are better than humans--number crunching type algorithms. The Demonstration Database algorithm is an example of work in that area. Through AI techniques, it may be possible to use some of the same ingenuity that humans use when they think about problems.

Despite the prevalent but erroneous belief that LISP or PROLOG (programming languages) are the only way to do AI or that if any program is written in those languages then it exemplifies AI, it is possible to use AI techniques in Ada, Pascal, or even FORTRAN. LISP does allow for easier data representation than FORTRAN (especially FORTRAN) nonetheless. Because of these reasons, initial coding will be done in FORTRAN (to meet military requirements) but future research will be done by this group with LISP and in particular the PC-Scheme dialect of LISP. Also being examined is possible purchase of an Ada compiler to use in AI coding.

What is an Expert System

One heavily researched area of AI is in Expert Systems. In writing computer code to solve problems in an area where humans historically make the intelligent decisions, it would be nice to use some of the same reasoning processes as those experts who presently solve such problems. This is the realm of Expert

---

[1]Elaine Rich, Artificial Intelligence, (New York : McGraw-Hill, 1983), p. 1.

Systems. Expert systems are usually developed through interviews with and observers of experts. Since, using the aforementioned techniques, it would be impossible to gather all of the "tricks of the trade", Expert Systems are not a complete representation of the expert.

Recognizing Patterns

One of the first areas in which AI will be applied to plane loading is in recognizing patterns in the items to be loaded. A pattern is defined as a matching among two or more items which (1) reduce the total square footage from that of haphazard loading of the same items, (2) significantly aid in balancing the aircraft. This section will discuss different ways in which AI could be used in this area.

Expert systems

Human load planners do a fairly good job of finding patterns in the long list of items they are given. They do this through many years of experience with similar lists of items. After a while, they can recall patterns which worked particularly well in the past. In the future, we would like to interview and examine some of the current load planners to give us a starting base of patterns which can help in the formulation of a "good" plane load.

How planes are currently loaded. In manual load planning, planners use a form which contains a scale diagram of the cargo floor of the plane to be loaded. Plastic templates of standard cargo units are manipulated on the scale diagram until a feasible and possibly an acceptable load plan is achieved. Not only does the load planner need to worry about the load fitting in the two dimensional plane but other requirements include (1) balance, (2) feasibility with respect to height, (3) hazardous cargo limitations, (4) position of cargo (priorities for loading and unloading), (5) total weight, and (6) weight per axle restrictions.[2] Knowledge of patterns which worked well in the past greatly helps the load planner in forming a good load in a time efficient manner.

Recognizing Existing Patterns

Assuming that pattern data can be collected and that a suitable data structure is formulated, it would then be necessary

---

[2]Douglas Cochard and Kirk A. Yost, "Improving Utilization of Air Force Cargo Aircraft", Interfaces, Vol 15, Number 1, (Jan-Feb 1985), pp.53-68).

to examine the current load requirement and recognize potential uses of the pattern database. There are many issues in doing this which make the problem difficult. One is the potential ambiguities about what items are in the load requirement. For example, how can a trailer be recognized if it is listed in the description field of the load requirement as "TRAILER", other times "TLR", and yet other times as "ACME TRLR" for some certain type of trailer. This issue is examined in section 4. Another problem is when a set of items can make different sets of patterns. In that case there must be a way to assess a worth to each pattern. This is the subject of the next paragraph.

Score algorithm. Some initial work was done in setting up a framework in which scores could be used as a measure of worth. A score algorithm could recognize all possible matchings of load units to patterns and then use some mathematical technique to "maximize" worth to the load plan. It should be mentioned that not all of the load units will be used in a pattern or that they should. Patterns may leave unfillable gaps in a plane, for example. Therefore it is emphasized that patterns are an aid to the plane loader and not a rule. AI techniques will help in analyzing tradeoffs between when to use or not use patterns.

Recognizing new patterns

The pattern world must adapt to new situations much like a human load planner would when his world changes. This section discusses issues in recognizing new patterns and the next examines identifying when it would be possible to make substitutions.

An area in which AI methods could be used is in the area of perceiving new configurations in the data which could help load plan in the future. Not only is it necessary to examine the past and the current load plans and write to memory all relationships with that data (such as what is touching what), but also to gleam from this mass of data the few patterns which may be useful in later load plans. Consequently, a basic need to having a program do this is to have a way to code human-like reasoning into such program. AI techniques will aid in doing this.

Recognizing acceptable replacements

Patterns are not, in general, concrete plans where no leeway is allowed. For example, one type of truck may do well in a situation if the type of truck mentioned in the pattern is not available. What classifies a "good" replacement is not strictly defined. An algorithm which can adapt to the current environment is a major area of research in Artificial Intelligence. One possibility of "telling" future runs of the load planner that a certain item can be replaced in some pattern is through an attached

"flag" which indicates this. This, nonetheless, is not the only way.

Balancing aircraft

As previously mentioned, it not only is necessary to find a feasible load with respect to the two dimensional floor plan, but is also must be asked "Will it fly?" AI may not be needed to calculate a center of balance, for example, in the circumstance where center of balances and weights are known for all constituent items loaded on the plane. But, this is not always the case. This data may be unknown or maybe variable due to different ways in which an item can be oriented. AI techniques are needed to substitute values when such problems exist given defaults, for example. As mentioned in the case of patterns, it may be hard to match up the item description to the item in the database which holds center of balance information.

Recognizing Flyability

However, balance and whether or not the plan fits are not the only determinations of flyability. Hazardous cargo has certain placement restrictions, certain items must be anchored in one of a few locations on an aircraft. Items may be restricted to only certain types of aircraft or maybe to one special plane which was outfitted especially for these items. Again, AI methods will help in making the program "reason" and "recognize."

Recognizing data set contents

Most of the above effort would be useless if it is impossible to let a computer "know" what items are sitting on the tarmac. Simply inputting 20 (or even fewer) character descriptions into a database doesn't mean the above requirement is satisfied. There are many different abbreviations and descriptions for the same item. A "dictionary" could be set up which lists abbreviations and either their single definition or a list of definitions and some computer recognizable description of in what contexts each definition would be accurate. Instructing a computer through a program that when you say "TRK CGO", generic truck cargo or maybe a cargo truck is what the item really is is demanding and can never cover all possible situations. Therefore, in addition to recognizing matches between description field and the algorithms database, it is also crucial that assumptions be made when above matches are not possible. Much AI research has been conducted in the area of Natural-Language Processing--the understanding of sentences. This area is not directly applicable to recognizing data set contents but may be extended somewhat.

Accomplishments

As previously mentioned, some additional coding has taken place in the area of the score algorithm. However, this effort is far from complete and little formulation has taken place in the other areas mentioned.

We have successfully developed an aircraft load planner using pattern matching where there is no scoring mechanism on the XEROX Dandelion Lisp workstations in the INTERLISP dialect of Lisp. Effort is currently being made to transform the algorithm to either an IBM PC or IBM 9375 compatible form. This effort is more fully described in the following section.

Aircraft Load Planning using Pattern Matching

As previously mentioned, patterns are a matching among items which aid in aircraft balance and reduce the amount of space required to load these items over the amount computed by simply adding together the square footage of the individual items. Using actual data from the TUCHA and SRF files, patterns were created in an ad-hoc manner--we made guesses about what cargo items go well together and what the attributes of a filled pattern are (length, width, and weight). For example, if a C5 is 228 inches wide and a cargo item is 80 inches wide and 150 inches long, simply adding together the widths would only allow two of those items to fit side by side. However if it is recognized in a pattern that three of these items could fit in a block 228 inches wide and 150 inches long, this would more accurately represent what is possible when the C5 is being loaded.

The algorithm loaded planes using an initial set of patterns and a list of cargo items (much reduced in scope and number from the cargo items in the TUCHA and SRF). Patterns were classified by the amount of plane they used (by length or weight) and were filled whenever possible with items from the cargo list. After all planes are loaded, the algorithm gleaned from the load plans possible candidates for new patterns. It then incorporated these new patterns in the permanent pattern database if the user thought they were important enough to learn. Therefore, this algorithm was able to learn and adapt through numbers of runs with different data.

Limitations As previously mentioned, there remains a significant problem in unambiguously recognizing which items are to be loaded. The algorithm did not allow for multiple representations for the same cargo. Another limitation is in the speed of the algorithm. Lisp is slow and will become unwieldy if the algorithm was run on a complete load requirement.

Current effort It is hoped that the algorithm can be translated into a form useable by MAC. Possible environments include PC/Scheme (a Lisp dialect) on IBM PC compatibles and Ada or Lisp on the IBM 9375. These possibilities are currently being explored.

Future priorities

It was mentioned in the section on recognizing data set contents that this was one area which was a prerequisite to the application of the other areas to the problem of plane loading. In spite of this, research and program coding can still be done in the other areas. Assumptions of unambiguous item descriptions can be made until the data set recognition code is finished. These assumptions will allow the following priorities. (1) Effort in the area of patterns in load planning included scoring algorithms will be continued. (2) Study of "expert" human load planners and their load plans will be done in order to formulate an initial pattern database. The problem of data set recognition will be an active research effort through the above levels despite its placement in the background.

PRINCIPAL INVESTIGATOR J J JARVIS CENTER NO. 243R6497OAO ACCOUNT NO. E-24-645

STATUS AT END OF JUNE 1988 DEPARTMENT I & S ENG

SPONSOR MARTIN MARIETTA

AWARD NUMBER 19X-SB778C RF CENTER NO. 00346073000 RESTRICTED FUND RF-49148

EFFECTIVE DATE 04-01-88 BILLING GROUP GTRC EXPIRATION DATE 12-31-88

	MONTH	FISCAL YEAR	TOTAL CONTRACT
PERSONAL SERVICES			
BUDGET			47,093.00
EXPENDED	7,997.46	7,997.46	7,997.46
ENCUMBERED	.00	.00	.00
FREE BALANCE			39,095.54
FRINGE BENEFITS			
BUDGET			10,971.00
EXPENDED	1,559.26	1,559.26	1,559.26
ENCUMBERED	.00	.00	.00
FREE BALANCE			9,411.74
MATERIALS AND SUPPLIES			
BUDGET			900.00
EXPENDED	.00	.00	.00
ENCUMBERED	.00	.00	.00
FREE BALANCE			900.00
TRAVEL			
BUDGET			3,750.00
EXPENDED	.00	.00	.00
ENCUMBERED	.00	.00	.00
FREE BALANCE			3,750.00
TOTAL DIRECT CHARGES			
BUDGET			62,714.00
EXPENDED	9,556.72	9,556.72	9,556.72
ENCUMBERED	.00	.00	.00
FREE BALANCE			53,157.28

GEORGIA INSTITUTE OF TECHNOLOGY

PRINCIPAL INVESTIGATOR J J JARVIS | CENTER NO. 243R64970AO | ACCOUNT NO. E-24-645

STATUS AT END OF JUNE 1988 | DEPARTMENT I & S ENG

SPONSOR MARTIN MARIETTA

AWARD NUMBER 19X-SB778C | RF CENTER NO. 00346073000 | RESTRICTED FUND RF-49148

EFFECTIVE DATE 04-01-88 | BILLING GROUP GTRC | EXPIRATION DATE 12-31-88

	MONTH	FISCAL YEAR	TOTAL CONTRACT	
OVERHEAD				
BUDGET			37,628.00	
EXPENDED	5,734.03	5,734.03	5,734.03	RATE OF 60.0 BASE OF 4
ENCUMBERED	.00	.00	.00	
FREE BALANCE			31,893.97	
TOTAL				
BUDGET			100,342.00	
EXPENDED	15,290.75	15,290.75	15,290.75	
ENCUMBERED	.00	.00	.00	
FREE BALANCE			85,051.25	

PROGRESS REPORT

Georgia Tech Aircraft Loading Project

July 1988

Activities Performed:

In July the effort in the PDRC centered on determining the issues and possible algorithms for dealing with level 4 data. During this month, weekly brainstorming sessions were used to flush out ideas and recognize possible trouble spots.

Algorithmic Ideas

The algorithms under consideration are modular, most involving three parts. These parts are a preprocessing stage, a first stage packing, and a plane packing stage. The preprocessor would palletize bulk cargo, if necessary, and perform sorting/arranging of the cargo and planes. The first stage packing would pack the cargo into strips of one plane width. It also would collect hazardous strips, and other utilization measures. This first stage allows the plane load packings to be performed as one dimensional (length), with a side constraint of total weight assigned to each aircraft. It also reduces the size of the problem to the number of strips generated instead of the number of cargo items. This stage could be implemented with a Next-Fit heuristic, or any other shelf or level oriented bin packing heuristic.

The plane packing stage, assigning strips to individual aircraft, could employ a multi-constraint knapsack heuristic to optimize the assignments. These heuristics rely on an almost greedy approach. Items are assigned based on a benefit to cost ratio. The choice of the benefit and cost functions are the determining factors in the speed and efficiency of the heuristics.[1] The use of this type of procedure would allow great flexibility in the objective of the solution methodology as a whole. Only the benefit and cost functions need be changed to alter the assignment criteria.

---

[1] For examples see Loulou and Michaelides, New Greedy-like Heuristics for the Multidimensional 0-1 Knapsack Problem, Operations Research, vol 27, no. 6, 1979.

Potential Problems

The following is a list of areas in which we would like clarification about our responsibilities, and the algorithm.

What is the Distinct algorithm?

In order that our methodology be easily incorporated into the Distinct scheduling routine, we must have a better understanding of what will be provided to our subroutine and what is required of its output.

Is our algorithm responsible for determining the aircraft type that is to be used? If so, can our algorithm choose a mix of aircraft types? Alternatively, will we be supplied with an aircraft type and an upper bound on the number of these aircraft available? If our algorithm decides that x number of aircraft are needed, will that be the number of aircraft sent, or can fewer than x aircraft be sent?

A first step in helping resolve some of these questions would be to send us some documentation on the Distinct algorithm. Directly related to these questions is the next.

What should our objective be?

We have thought of two possible types of objective criteria for our packing routine. The first is to minimize the number of planes required to lift all of the cargo on the list. Variants of this objective could limit the number of aircraft types available, and their individual numbers.

A competing objective would be to maximize the value of each plane load. This could be implemented with pre-assigned aircraft types or determining types by matching gross cargo specifications to aircraft capabilities. This objective would weight items by arrival priority, and try to place items of similar priority on the same plane.

The second objective criteria is better suited to a decision process where it is not necessary to move all of the cargo. Hence, plane loads are prioritized so if there are not a sufficient number of planes available, those loads with the highest priority may be chosen. In order to do this, however, we would need priorities on the level 4 items which are not available in the SRF or TUCHA.

Are the aircraft configured?

When dealing with CRAF is the configuration fixed? This also involves the previous question of whether we choose the aircraft types or they are given to us. If they are given to us, can we change the configuration? If we must choose the aircraft types then

each configuration must be considered as a different aircraft type, i.e. a list of attributes (cargo bay dimensions, number of PAX and ACL). This will explode the number of types that our algorithm will have to consider in matching aircraft types to cargo.

Speed and Size

Any guidelines on the desired speed of the algorithm for a given size of problem would be helpful. What is the number of level 4 records and individual items that will be passed in one call to our subroutine (average and maximum)? What is an acceptable time limit for the average size data set?

DATA

Square Feet

In some of the level4 records the square feet indicated does not coincide with the reported length and width of the item. In some cases the two measures differed by as much as 5 sq. ft. Which of these measures, reported square feet or length x width, should take precedence?

Cargo Category Codes

The two category codes, E and K, from Appendix T table 18, confuse security and hazardous cargo. This may force any item from one of these code classes to be considered hazardous, and packed as such.

Hazardous Cargo

We do not know how to handle hazardous cargo. While we realize that it usually must be placed near an exit, how much hazardous cargo can be placed on one aircraft? Also, are there incompatible hazardous cargo, and if so how can they be detected based on the cargo category codes?

Obtaining a copy of AFR 71-8/TM 38-236, Preparation of Hazardous Cargo for Military Air Shipment, for our use may resolve some of these questions.

Input/Output

What will be the I/O requirements of Distinct's algorithm with respect to our subroutine (exact format specifications)? Will items be given as SRF/TUCHA records? For hazardous cargo, more information than is supplied in a SRF record may be needed.

How will aircraft data be supplied? Will ACL's be adjusted for critical leg fuel requirements? Will aircraft be partially loaded upon arrival?

Conclusion

All of the points brought up in this letter are important and need to be resolved as soon as possible, so that a properly designed, practically useful algorithm can be developed with as few revisions as possible.

PRINCIPAL INVESTIGATOR J J JARVIS CENTER NO. 246R6497OAO ACCOUNT NO. E-24-645

STATUS AT END OF JULY 1988 DEPARTMENT I & S ENG

SPONSOR MARTIN MARIETTA

AWARD NUMBER 19X-SB778C RF CENTER NO. 00648073000 RESTRICTED FUND RF-49148

EFFECTIVE DATE 04-01-88 BILLING GROUP GTRC EXPIRATION DATE 12-31-88

	MONTH	FISCAL YEAR	TOTAL CONTRACT
PERSONAL SERVICES			
BUDGET			47,093.00
EXPENDED	5,754.90	5,754.90	13,752.36
ENCUMBERED	25,297.71	25,297.71	25,297.71
FREE BALANCE			8,042.93
FRINGE BENEFITS			
BUDGET			10,971.00
EXPENDED	1,076.84	1,076.84	2,636.10
ENCUMBERED	2,153.68	2,153.68	2,153.68
FREE BALANCE			6,181.22
MATERIALS AND SUPPLIES			
BUDGET			900.00
EXPENDED	304.00	304.00	304.00
ENCUMBERED	64.00	64.00	64.00
FREE BALANCE			532.00
TRAVEL			
BUDGET			3,750.00
EXPENDED	.00	.00	.00
ENCUMBERED	.00	.00	.00
FREE BALANCE			3,750.00
TOTAL DIRECT CHARGES			
BUDGET			62,714.00
EXPENDED	7,135.74	7,135.74	16,692.46
ENCUMBERED	27,515.39	27,515.39	27,515.39
FREE BALANCE			18,506.15

PRINCIPAL INVESTIGATOR J J JARVIS CENTER NO. 246R6497OAO ACCOUNT NO. E-24-645

STATUS AT END OF JULY 1988 DEPARTMENT I & S ENG

SPONSOR MARTIN MARIETTA

AWARD NUMBER 19X-SB778C RF CENTER NO. 00646073000 RESTRICTED FUND RF-49148

EFFECTIVE DATE 04-01-88 BILLING GROUP GTRC EXPIRATION DATE 12-31-88

	MONTH	FISCAL YEAR	TOTAL CONTRACT	
OVERHEAD				
BUDGET			37,628.00	
EXPENDED	4,281.44	4,281.44	10,015.47	RATE OF 60.0 BASE OF 4
ENCUMBERED	16,509.23	16,509.23	16,509.23	
FREE BALANCE			11,103.30	
TOTAL				
BUDGET			100,342.00	
EXPENDED	11,417.18	11,417.18	26,707.93	
ENCUMBERED	44,024.62	44,024.62	44,024.62	
FREE BALANCE			29,609.45	